

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

GIRAFA.COM, INC.,)	
)	
Plaintiff,)	
)	
v.)	
)	
)	C.A. NO. 07-787 (SLR)
AMAZON WEB SERVICES LLC,)	
AMAZON.COM, INC., ALEXA)	
INTERNET, INC., IAC SEARCH &)	
MEDIA, INC., SNAP TECHNOLOGIES,)	
INC., YAHOO!, INC., EXALEAD S.A.,)	
and EXALEAD, INC.,)	
)	
Defendants.)	
)	

**DECLARATION OF MARK D. NIELSEN IN OPPOSITION TO GIRAFA.COM'S
MOTION FOR PRELIMINARY INJUNCTION**

I, Mark D. Nielsen, declare:

1. I am an attorney licensed to practice in the States of California and Texas and before the United States Courts of Appeals for the Ninth and Federal Circuits, as well as the Central, Eastern, Northern, and Southern Districts of California, and Northern and Southern District of Texas. I am also licensed to practice before the United States Patent and Trademark Office. I am admitted pro hac vice before this Court for this case. I am a partner in the law firm of Cislo & Thomas LLP representing Plaintiff Snap Technologies, Inc. ("Snap"). I make this declaration of my own personal knowledge or on information and belief where so stated. If called as a witness, I could and would competently testify to the truth of the matters asserted herein.

2. The letter order of the exhibits in this declaration begins at Exhibit B to avoid confusion with the concurrently submitted Agostino Declaration, which contains an Exhibit A.

3. Attached hereto as Exhibit B is a true and correct copy of an article entitled “Web Representation with Dynamic Thumbnails” by Stephan Schmid. This article was published in March of 1998, as indicated in other patent documents, such as United States Patent 6,301,607.

4. Attached hereto as Exhibit C is a true and correct copy of United States Patent 7,177,948 to Kraft.

5. Attached hereto as Exhibit D is a true and correct copy of United States Patent 6,496,206 to Mernyk.

6. Attached hereto as Exhibit E is a true and correct copy of United States Patent 6,594,697 to Praitis.

7. Attached hereto as Exhibit F is a true and correct copy of an article entitled “Visual preview for link traversal on the World Wide Web” by Theodorich Kopetzky and Max Muhlhauser. This article was published in May of 1999 as indicated at the URL http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VRG-405TDWC-15&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=e97180017d913c62f65cf18e15f12322.

8. Attached hereto as Exhibit G is a true and correct copy of United States Patent 6,108,703 to Leighton.

9. Attached hereto as Exhibit H is a true and correct copy of United States Patent 5,991,809 to Kriegsman.

10. Attached hereto as Exhibit I are true and correct copies of the referenced pages from the transcript of the April 18, 2008 deposition of Brad A. Myers that Snap cited in its memorandum in opposition to Girafa's motion for preliminary injunction.

11. Referenced hereto as Exhibit J, but filed in a separate document under seal, are true and correct copies of pages from the transcript of the April 9, 2008 deposition of Shirli Ran that Snap cited in its memorandum in opposition to Girafa's motion for preliminary injunction. This exhibit has been filed under seal based on Girafa's request to maintain the deposition transcript as confidential.

I declare under penalty of perjury that the foregoing is true and correct.

Executed on May 16, 2008 at Santa Monica, California.



Mark D. Nielsen

EXHIBIT B

Web Representation with Dynamic Thumbnails

Stefan Schmid¹,
Department of Distributed Systems,
University of Ulm, Germany

Abstract: The popularity of the *World Wide Web (WWW)* has led to rapid growth of Web sites all over the world. Thousands of new Web pages are designed every day. Today, Web pages with embedded hyper-links rendered by Web browsers are only weak representation of the Web topology. Using static thumbnails (small images) of Web pages to represent relations of Web pages is an employed technique. Due to the limitations of permanent images, we propose a novel online service to provide up-to-date thumbnails of any Web pages. Online provided and dynamically generated thumbnails open new ways to represent the Web and enhance Web designers potentialities. As an application, we show a VRML-based user interface that visualizes a user's vicinity while browsing the Web.

1 Introduction

Visualizing the hardly structured topology of the Web requires sophisticated representation techniques. The Web consists merely of Web pages, each of which has its unique address called *Uniform Resource Locator (URL)* as nodes and hyper-links, named *Hypertext References (HREF)*, to other Web pages as interconnection between nodes. Small areas of the Web are presented to users by Web browsers.

In order to enhance Web representation, we suggest to improve the power of expression of hyper-links. Text-only links, if well chosen, usually give information about the page content. Thumbnail links, where the image represents a miniature of the referenced Web page, contain information of the page layout and the content. Combining both techniques the user may perceive a maximum of information regarding content and layout. See figure 1 as an example.

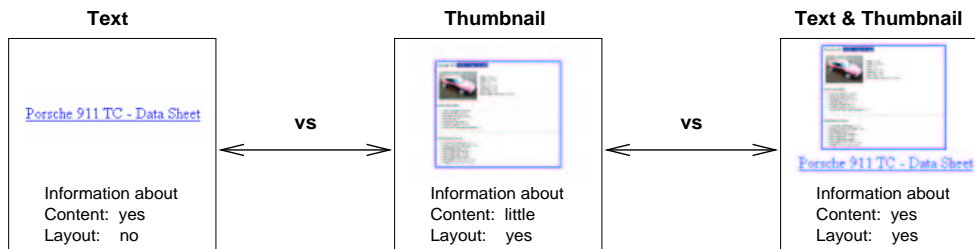


Figure 1: Textual links together with thumbnails contain information on both content and layout of linked pages

2 Static Thumbnails

Static thumbnails are pre-rendered and reduced-size images that point to the large original picture [1]. As a small representation of a larger image, they provide a quick preview of the content of a picture. The reader can click on the miniature to view the full-size graphic.

So far, Web designers use thumbnails mainly to shorten download time and to reduce data load. A thumbnail shows a small preview to attract the reader's interest without delaying the download like a full-size graphic. They are often applied to Web pages that give the reader an overview, e.g. in virtual galleries² or content tables.

¹ Student Research Assistant, Email: sschmid@hydra.informatik.uni-ulm.de

² Sites where many pictures and graphics are exhibited

Furthermore, Web designers often use images and thumbnails to enhance the design of their pages. Readers usually prefer Web pages with graphics if there are no bandwidth constraints.

Besides the advantages of static thumbnails, they surely have negative aspects. First of all, it is expendable work to create good-looking and meaningful images manually. Secondly, static thumbnails are often outdated. When Web sites are extended or existing pages are changed, thumbnails have to be re-made. In the next section, we propose dynamic thumbnails — an effort to overcome the limitations of static thumbnails.

3 Dynamic Thumbnails

Dynamic thumbnails are online provided and on demand rendered images of Web pages. Generating thumbnails dynamically requires online accessible sources of the original representation. Since Web page thumbnails enhance hyper-link representation (see figure 1) and moreover Web pages are online accessible through Web servers, we chose them as sources for our thumbnails.

3.1 General Architecture

The general architecture of the *Thumbnail Service* is based on a client-server model. Clients, such as ordinary Web browsers, Java applets, VRML programs [2], or other stand-alone programs, access the Thumbnail Server via the Internet protocol TCP/IP.

A client requests thumbnails of arbitrary Web pages. After the request is received, the server first checks its memory and disk caches. If the requested Web page thumbnail is already rendered and not expired, it is immediately sent back to the client. Otherwise, the server loads the requested Web page and renders the image. There are several possible ways to achieve the render process. In our implementation, we use an existing Web browser (e.g. Netscape) to render the page. Then, the server captures the rendered image and scales the bitmap according to width and height of the client request. Afterwards, the image is encoded according to JPEG [5]. The new generated thumbnail is stored in the server's caches and sent in reply to the client. In order to provide thumbnails concurrently to multiple clients, the server must be capable to process thumbnails simultaneously. This procedure is explained in the full paper.

Figure 2 shows how clients, the Thumbnail Server, the rendering engine, and Web servers or proxies are interrelated.

3.2 Server-Client Protocol

The Thumbnail Server and the client applications use the *Hyper Text Transfer Protocol (HTTP)* as communication protocol.

The server acts like a HTTP server regarding its clients. Incoming client requests are parsed to extract the request parameters such as URL, WIDTH, HEIGHT, etc. of Web page thumbnails. The HTTP protocol allows users to request thumbnails directly from the Web browser according to RFC/1945 [4]. That offers the opportunity to simply integrate dynamically generated thumbnails into Web pages. Neither are specific client applications nor Web browser plug-ins necessary - merely a Web browser is required as a client. Alternatively, self-implemented client programs such as Java programs, browser plug-ins, cgi-scripts, servlets etc. can be used to access the server.

The request format is designed to be HTTP compliant. Each request is encoded as an HTTP-GET [4] request. The general request syntax (in EBNF notation) is as follows:

```
request ::= http://<server name>[:<port>]/?url=<url>{%<options>}
```

3.3 Web Page Rendering

Our implementation uses the current Netscape version installed on the server host as rendering engine. The browser is remotely controlled by Microsoft's Dynamic Data Exchange (DDE) interprocess communication system.

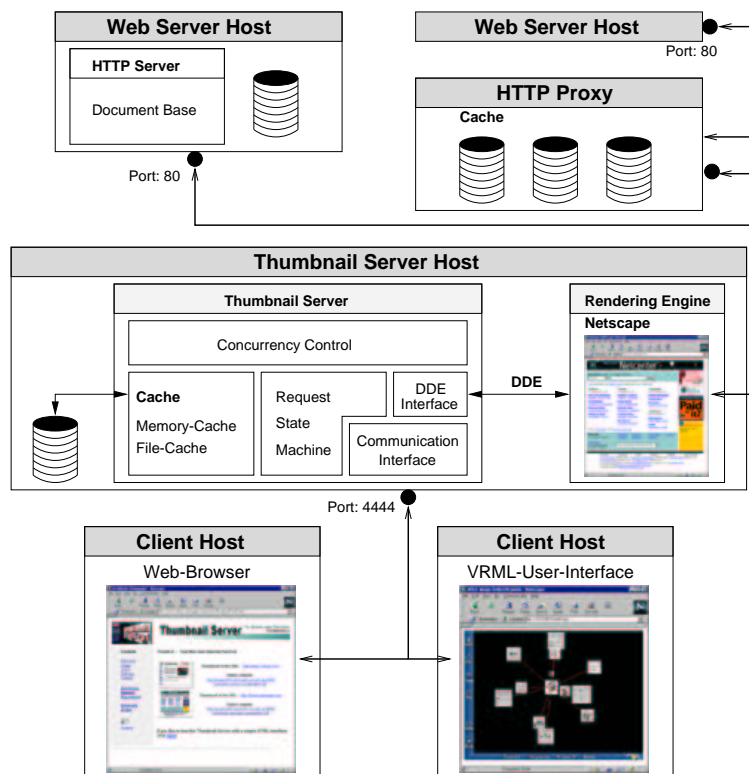


Figure 2: Thumbnail Server Architecture

Using an external Web browser as a rendering engine implies that the software depends on third party applications. In spite of this disadvantage, there are good reasons to live with this restriction. First, we saved considerable time of re-implementing a rendering engine. Second, due to remote controlling a major Web browser such as Netscape, we always have an up-to-date Web page rendering engine. No matter what new HTML features (e.g. frames, layers) are developed, we do not have to adjust our rendering engine. And third, the thumbnails look like users commonly perceives the Web page — good or bad.

4 An Application: The VRML User Interface of CoBrow

During the development period of the CoBrow³ project, we needed a representation of a user's vicinity while browsing the Web. Talking about a user's vicinity, we mean the user's neighbors⁴ and the Web pages closely linked to the page the user reads. The solution was to use a web of images consisting of user icons (representing neighbors) and Web page thumbnails (representing Web pages near the user's location) [3]. The need of a new service which provides dynamically generated thumbnails of arbitrary Web pages became obvious. Figure 3 presents a screen shot of the CoBrow VRML user interface using the Web page thumbnails.

5 Summary

Representing Web pages and other large graphics, “thumbnailing” is already a commonly used technique by Web designers. We have discussed that Web page thumbnails in interrelation with textual links increase the reader's knowledge of content and layout of referenced Web pages.

³CoBrow is a project in the EU “Telematics for Research” research program line. More information are available at <http://www.cobrow.com/>

⁴People who read the same Web page or browse the same area of the Web

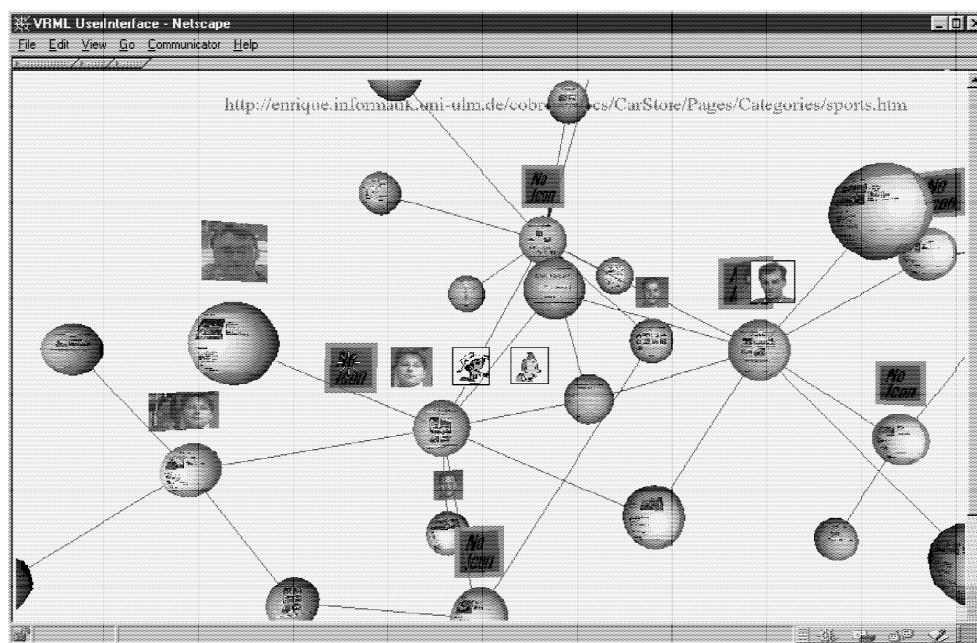


Figure 3: CoBrow VRML User Interface

Therefore, employing thumbnails in Web design improves Web sites. The Thumbnail Server, as a new Internet service, allows the use of thumbnails even in dynamic environments. As a result, thumbnails, generated and provided “on the fly”, enable new representations of the Web. The application of thumbnails in our VRML-based Web user interface provides the opportunity to experience the Web in a novel fashion.

6 Further Work

Believing that thumbnails improve Web representation and enable new Web visualization techniques, we would like to standardize the Thumbnail service. Our goal is to have a reliable Thumbnail service on each Web site responsible for the latest thumbnails. We also make best effort to find new representation models of the Web based on virtual reality and improved by thumbnails. Besides that, we continue to enhance the system and the service provided to users.

References

- [1] L.W. Arrington, *Understanding Thumbnail Images*, Montgomery County Public School, 1996
online at: URL: <http://www.bev.net/education/schools/admin/thumbnails.html>
- [2] Information technology: *Computer graphics and image processing – The Virtual Reality Modeling Language (VRML)*, International Standard ISO/IEC 14772-1:1997
- [3] H. Bönisch, S. Fiedler, K. Froitzheim, P. Schulthess, *A VRML-based Visualization of User-Vicinities in the WWW*, 1997
- [4] T. Berners-Lee, R. Fielding, H. Frystyk, *RFC/1945: Hypertext Transfer Protocol – HTTP/1.0*, May 1996, online at: URL: <http://www.roxen.com/rfc/rfc1945.html>
- [5] ISO DIS 10918-1., *Digital Compression and Coding of Continuous-tone Still Images (JPEG)*, CCITT Recommendation T.81.

EXHIBIT C



US007177948B1

(12) **United States Patent**
Kraft et al.

(10) **Patent No.:** **US 7,177,948 B1**

(45) **Date of Patent:** **Feb. 13, 2007**

(54) **METHOD AND APPARATUS FOR
ENHANCING ONLINE SEARCHING**

(75) Inventors: **Reiner Kraft**, Gilroy, CA (US);
Neelakantan Sundaresan, San Jose,
CA (US)

(73) Assignee: **International Business Machines
Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/442,150**

(22) Filed: **Nov. 18, 1999**

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/246; 707/3; 715/800;**
715/801

(58) **Field of Classification Search** 345/704,
345/738, 968, 739, 709, 918; 707/5, 3; 709/203,
709/223, 224, 205, 273, 217-219, 246; 705/27;
715/700, 800, 801

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,708,825 A * 1/1998 Sotomayor 715/501.1
5,764,235 A * 6/1998 Hunt et al. 345/428
5,842,206 A * 11/1998 Sotomayor 707/2
5,847,708 A * 12/1998 Wolff 715/764
5,870,549 A * 2/1999 Bobo, II 709/206
5,870,770 A * 2/1999 Wolfe 715/805
5,877,766 A * 3/1999 Bates et al. 715/854
5,884,056 A * 3/1999 Steele 715/738
5,933,140 A * 8/1999 Strahorn et al. 345/709
5,963,969 A * 10/1999 Tidwell 707/500
5,982,369 A * 11/1999 Sciammarella et al. 345/835
5,986,654 A * 11/1999 Alexander et al. 345/744
6,006,265 A * 12/1999 Rangan et al. 709/226
6,070,176 A * 5/2000 Downs et al. 345/848
6,226,655 B1 * 5/2001 Borman et al. 715/501.1

6,269,362 B1 * 7/2001 Broder et al. 707/1
6,272,484 B1 * 8/2001 Martin et al. 707/1
6,301,586 B1 * 10/2001 Yang et al. 707/104.1
6,307,573 B1 * 10/2001 Barros 345/440
6,335,746 B1 * 1/2002 Enokida et al. 345/839
6,405,192 B1 * 6/2002 Brown et al. 707/3
6,567,177 B2 * 5/2003 Matsuyama 358/1.14

OTHER PUBLICATIONS

Czerwinski et al., The Contributions of Thumbnail image, Mouse-
over Text and Spatial Location Memory to Web Page Retrieval in
3D, May 1999, *Interact '99*, pp. 163-170.*

Microsoft Press Computer Dictionary, Third Edition, 1997, p. 156.*
<http://www.camcity.com> (Apr. 7, 1999).*

Dragutsky, "Image search Engine Reviews: Scour.Net, GIF Wizard
and Image Surfer", *Search and Metasearch Engines*, Oct. 2, 1998.*

Dragutsky, "Image Search Engine Reviews: Scour.Net, GIF Wizard
and Image Surfer", *Search and Metasearch Engines*, Oct. 2, 1998.

* cited by examiner

Primary Examiner—Saleh Najjar

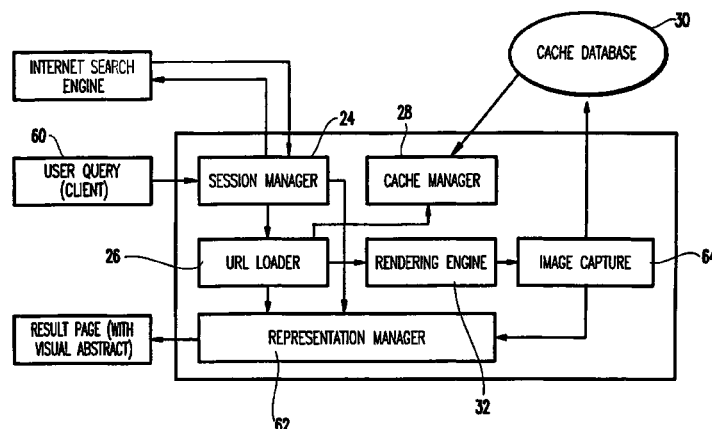
Assistant Examiner—Dhairya A. Patel

(74) *Attorney, Agent, or Firm*—Edmund Mizumoto, Esq.;
McGinn IP Law Group, PLLC

(57) **ABSTRACT**

A method and apparatus are provided for processing search
results obtained in response to a user query. This may
include examining document pointers returned by a search
engine to identify a source from which the documents are
available and generating at least two visual abstracts of each
of the documents. Each of the visual abstracts may be of a
different size. The method and system may also include
formatting a stream of data such that when the data is
displayed, a smaller one of the visual abstracts appears
adjacent to a corresponding search result.

24 Claims, 8 Drawing Sheets



U.S. Patent

Feb. 13, 2007

Sheet 1 of 8

US 7,177,948 B1

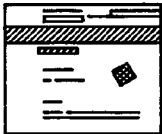
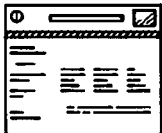
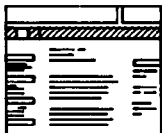
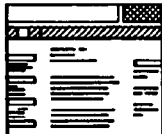
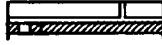
OTHER XML RESOURCES	
<p>1) http://www.infoworld.com/cgi-bin/display/commerce.pl?prophet.htm Rank 10%: Abstract: Market's love affair with Internet stocks won't end happily (InfoWorld) START DC Enhanced CODE Iframe for no Iframes Layers Capable Browsers Ifayer The Layer definition is located at the bottom of the</p>	
<p>2) http://www.infoseek.com/ Rank 10%: Abstract: GO Network-Start Here- Header Begin Go branding and searchbox placement Member service links. Member Services New Membership Free-E-mail sign in BEGIN SEARCH INFOSEEK SEARCH Tips Advancedsearch START</p>	
<p>3) http://www.wired.com/news/news/culture/story/10124.html Rank 8%: Abstract: Culture News from Wired News TOP AD BANNER WIRED NEWS header WIRED NEWS header updated5:55p.m.20.Jan.99.PST Wired News Webmoneky Web 101 HotWired Archives Wired Magazine Suck.com The Web--HotBot end</p>	
<p>4) http://www.wired.com/news/news/technology/story/16221.html Rank 8%: Abstract: Technology News from Wired News TOP AD BANNER WIRED NEWS header WIRED NEWS header updated5:55p.m.20.Jan.99.PST Wired News Webmonkey Web 101 HotWired Archives Wired Magazine Suck.com The Web--HotBot e</p>	
<p>5) http://www.wired.com/news/news/technology/story/12888.html</p>	

FIG.1

5

U.S. Patent

Feb. 13, 2007

Sheet 2 of 8

US 7,177,948 B1

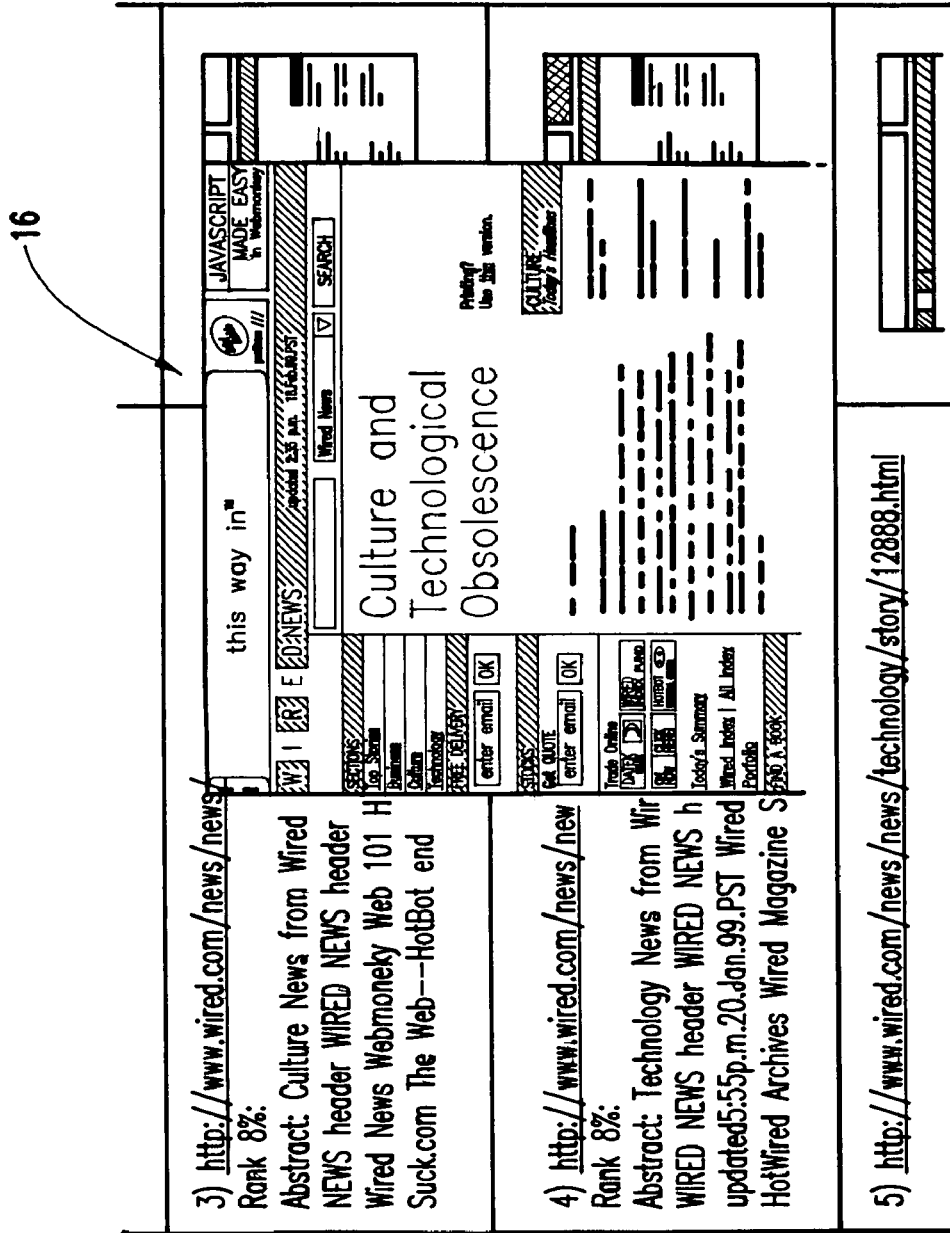


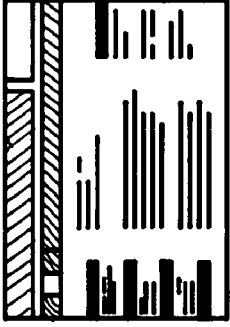
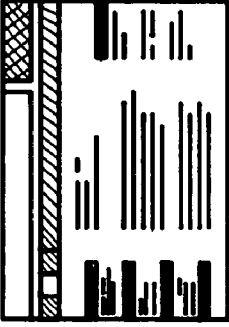

FIG.2

U.S. Patent

Feb. 13, 2007

Sheet 3 of 8

US 7,177,948 B1

<p>3) http://www.wired.com/news/news/culture/story/10124.html Rank 8%: Abstract: Culture News from Wired News TOP AD BANNER WIRED NEWS header WIRED NEWS header updated5:55p.m.20.Jan.99.PST Wired News Webmoney Web 101 HotWired Archives Wired Magazine Suck.com The Web---HotBot end</p>	
<p>4) http://www.wired.com/news/news/technology/story/16221.html Rank 8%: Abstract: Technology News from Wired News TOP AD BANNER WIRED NEWS header WIRED NEWS header updated5:55p.m.20.Jan.99.PST Wired News Webmoney Web 101 HotWired Archives Wired Magazine Suck.com The Web---HotBot e</p>	
<p>5) http://www.wired.com/news/news/technology/story/12888.html</p>	

10

12

FIG.3

U.S. Patent

Feb. 13, 2007

Sheet 4 of 8

US 7,177,948 B1

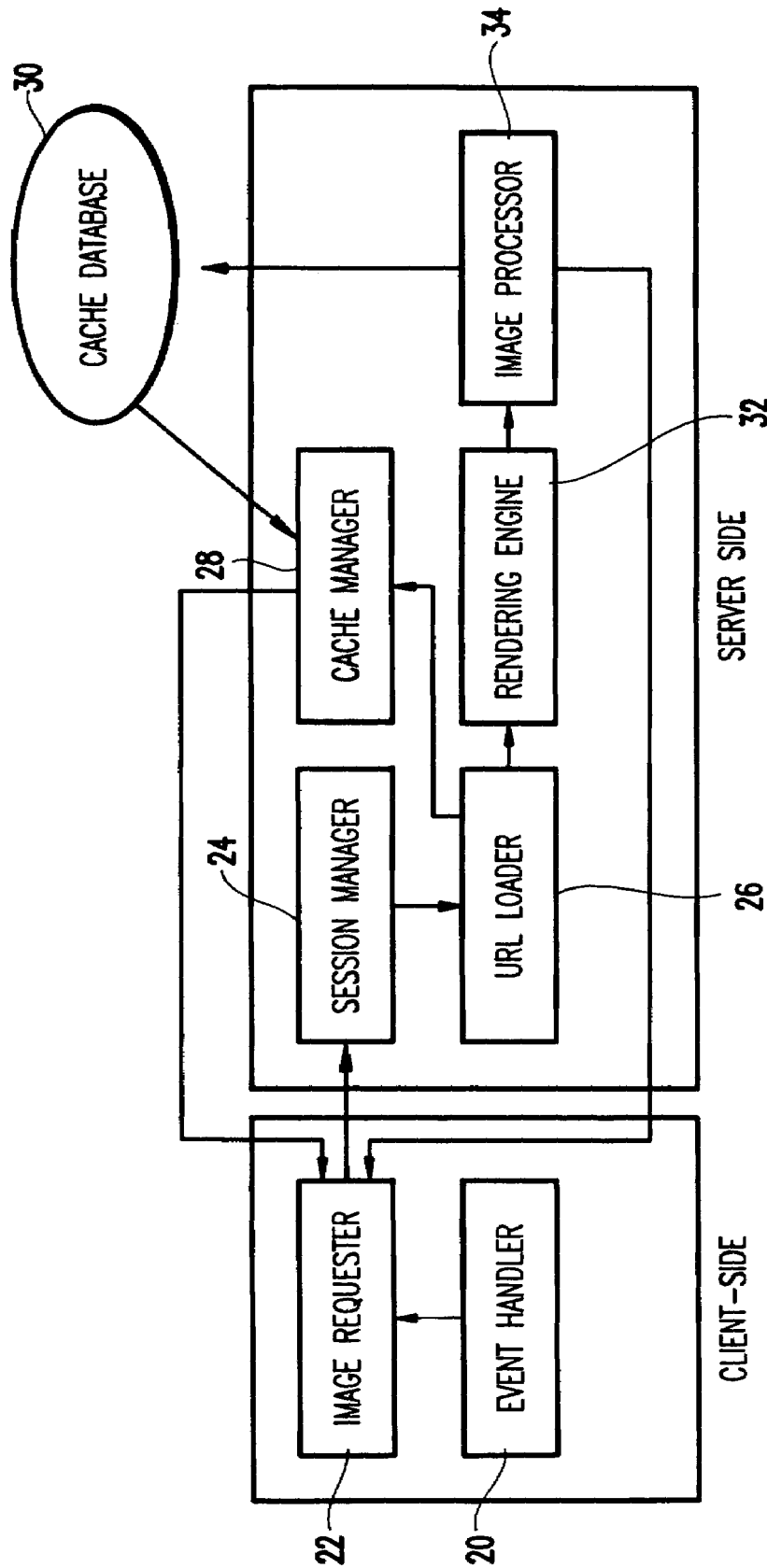


FIG.4

U.S. Patent

Feb. 13, 2007

Sheet 5 of 8

US 7,177,948 B1

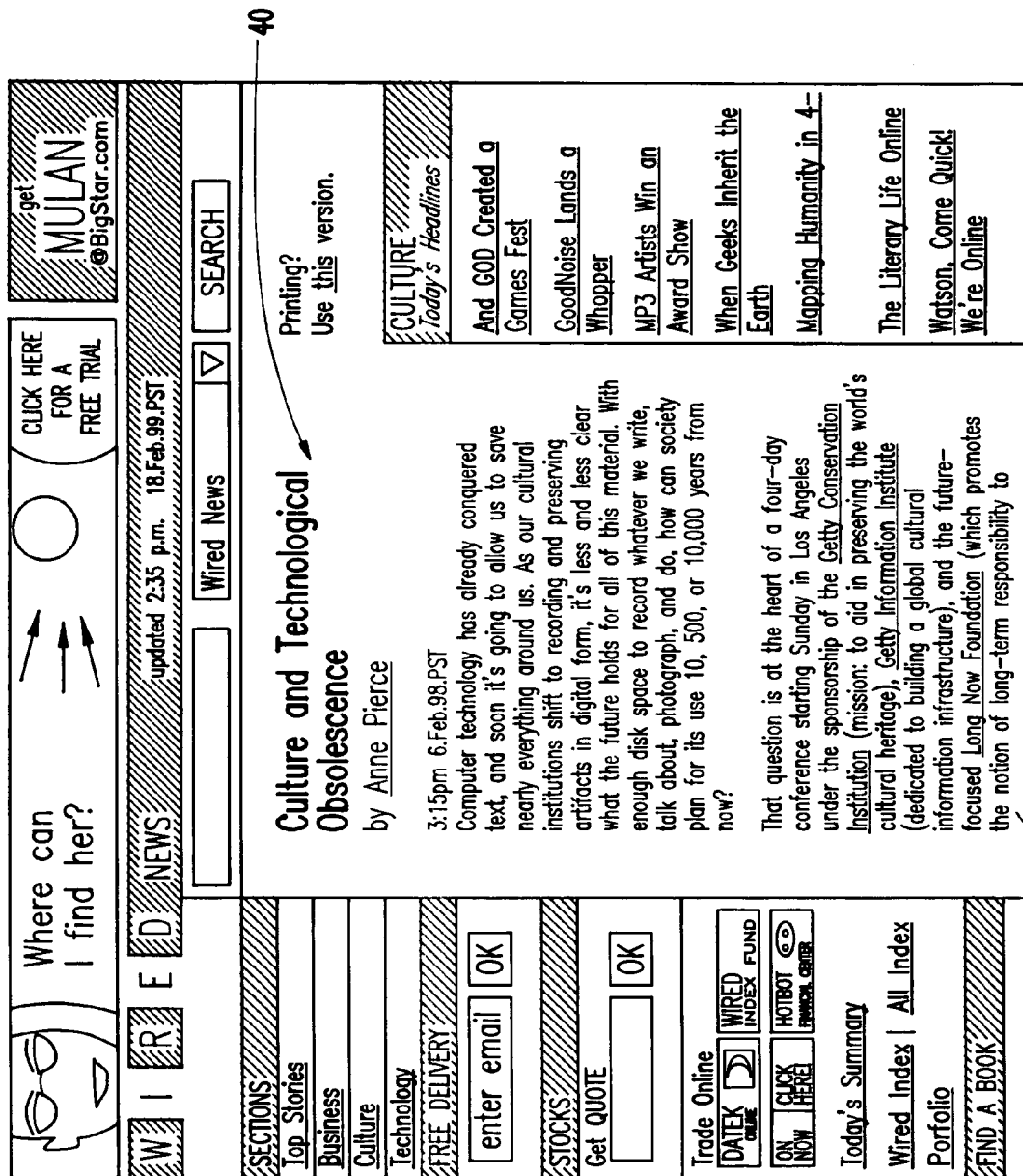


FIG.5

U.S. Patent

Feb. 13, 2007

Sheet 6 of 8

US 7,177,948 B1

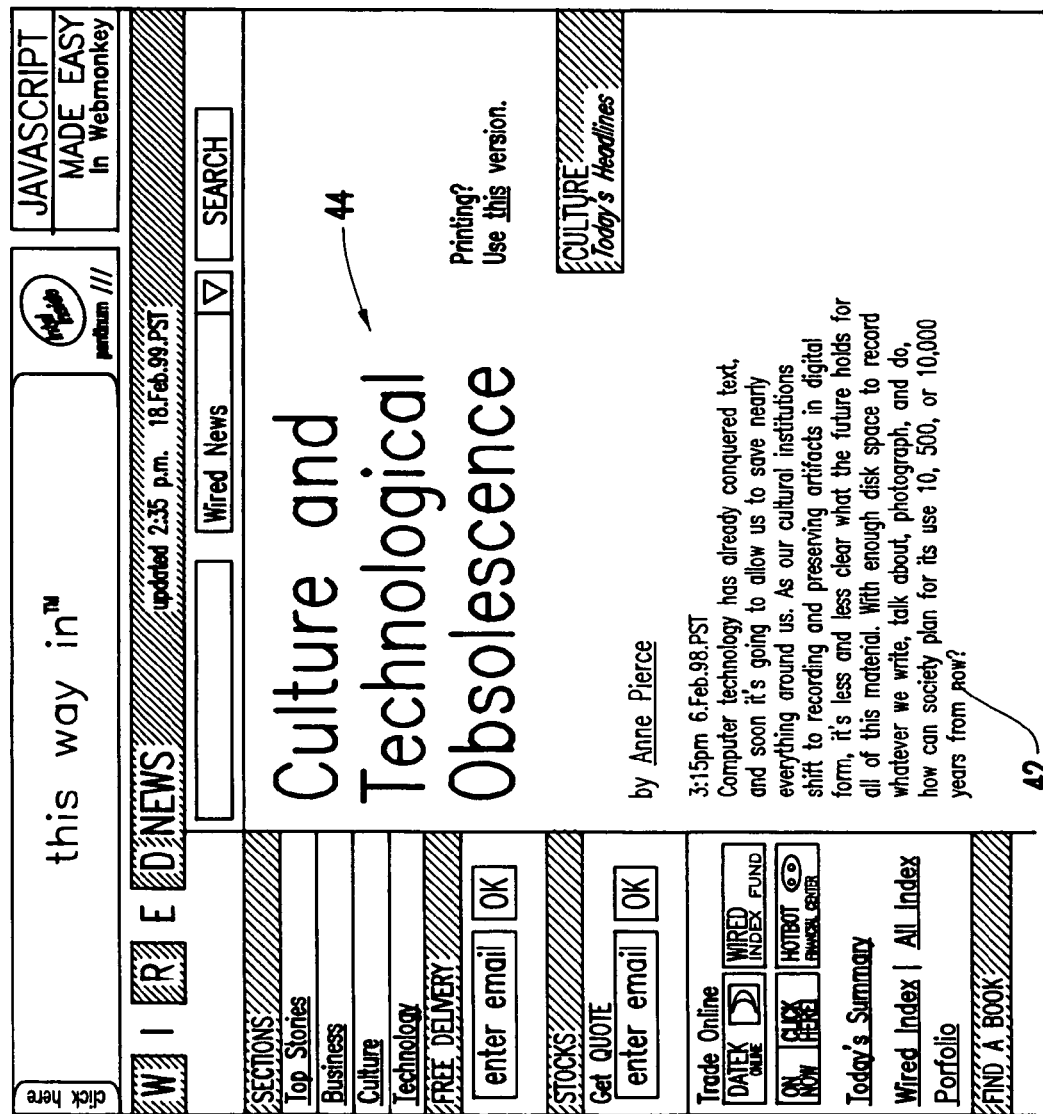


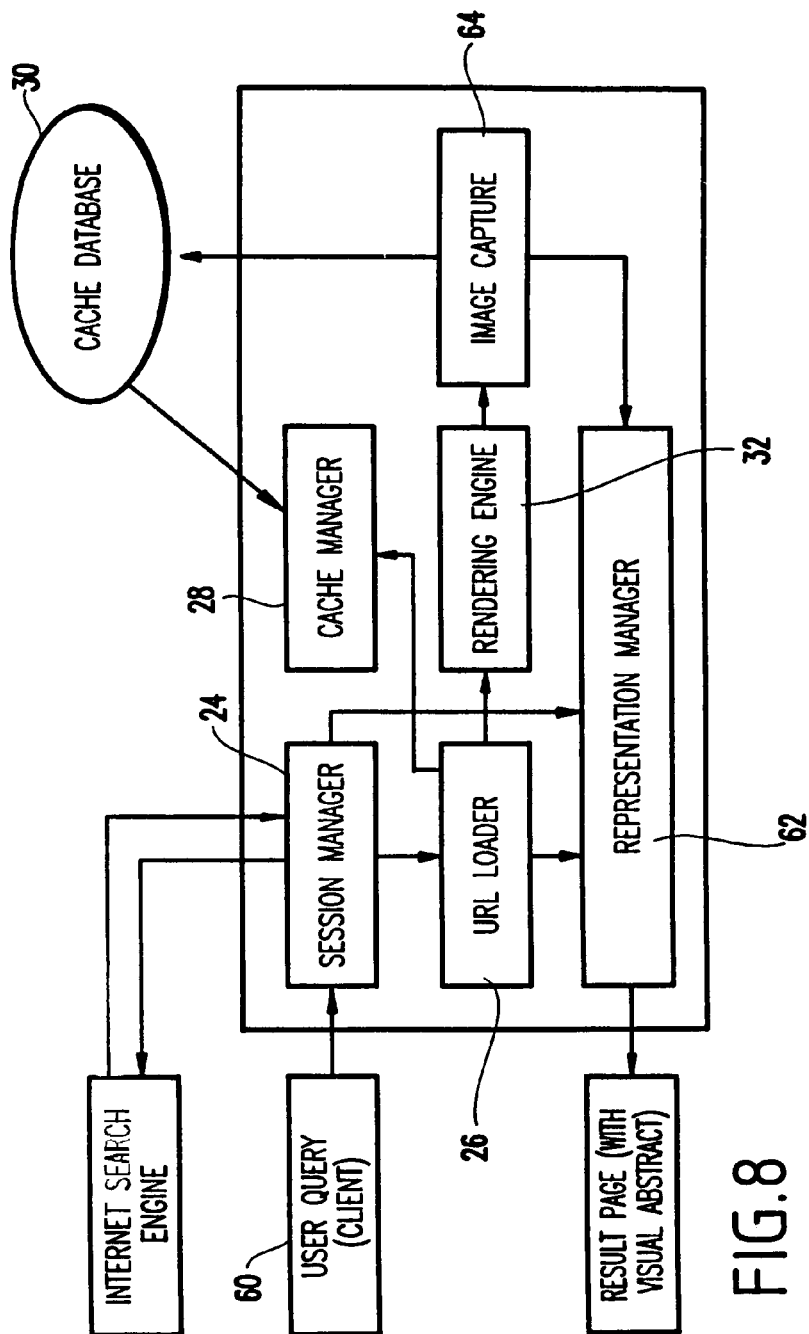
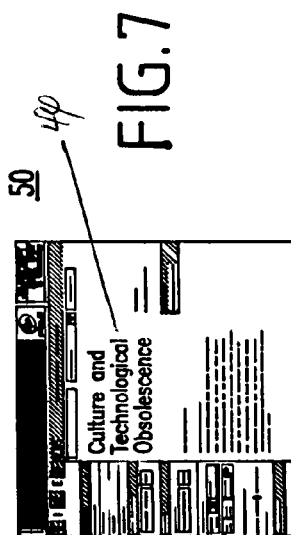
FIG. 6

U.S. Patent

Feb. 13, 2007

Sheet 7 of 8

US 7,177,948 B1

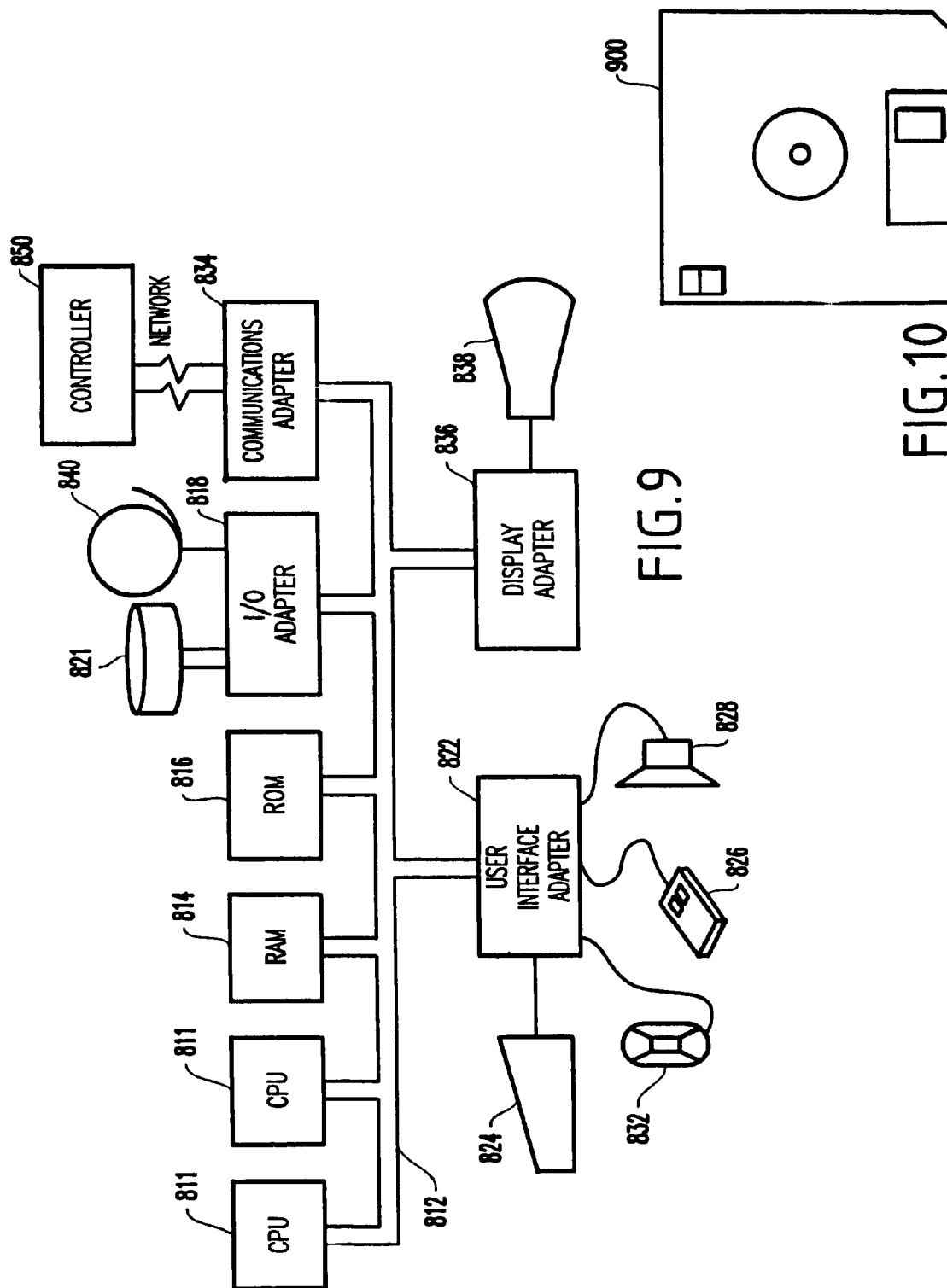


U.S. Patent

Feb. 13, 2007

Sheet 8 of 8

US 7,177,948 B1



US 7,177,948 B1

1

**METHOD AND APPARATUS FOR
ENHANCING ONLINE SEARCHING****BACKGROUND OF THE INVENTION****1. Field of the Invention**

The present invention generally relates to online search technologies and document summarizations. More specifically, the present invention relates to a method and apparatus for efficiently processing search results obtained in response to a user query.

2. Description of the Related Art

An important use of computers is the transfer of information over a network. Currently, the largest computer network in existence is the Internet, which, as is well known, is a worldwide interconnection of computer networks that communicate using a common protocol. Millions of computers, from low end personal computers to high end super computers, are connected to the Internet.

In the late 1980s, a new type of information system, known as the World Wide Web ("the Web") was introduced to the Internet. As is well known, the Web is a wide-area hypermedia information retrieval system aimed to give wide access to a large universe of documents.

The architecture of the Web follows a conventional client-server model. The terms "client" and "server" refer to a computer's general role as a requester of data (i.e., the client) or a provider of data (i.e., the server). In the Web environment, Web browsers are clients and Web documents reside on servers. Web clients and Web servers communicate using a protocol called "Hypertext Transfer Protocol" (HTTP). A browser opens a connection to a server and initiates a request for a document. The server delivers the requested document, typically in the form of a text document coded in a standard Hypertext Markup Language (HTML) format.

Portions of documents displayed on the Web may contain hypertext links. The hypertext links link graphics or text on one document with another document on the Web. Each hypertext link is associated with a Universal Resource Locator (URL). A URL specifies a server and a particular document on that server. When a user selects a hypertext link, using, for instance, a cursor, the browser connects to the server and retrieves the document(s) specified by the URL(s).

Some servers provide a means for searching a collection of documents. Upon initial request, the server supplies a form to the browser. The user, using the browser, enters data such as keywords on this form as part of a search query and then opens a new connection to the server and submits this data to the server. The server responds to this request with a new document listing, some or all of the documents matching those key words or other data requested by the browser. Each listed document normally includes a hypertext link to the actual document so that the user may easily retrieve that document.

Today, finding information as easily and quickly as possible has become a crucial problem. The World Wide Web contains millions of documents spread over hundreds of thousands of computers throughout the world. Although hypertext links tie all these documents together, the distributed architecture of the Web produces an incoherent system that often makes it very difficult for users to locate documents of interest.

Search engines have become more and more important with the continuous growth of information in order to find and retrieve information from a large repository such as the Internet and databases. As is well known, current search

2

technology is usually based on an electronic search form, where the user enters keywords to form a query. As discussed above, the query is submitted to the search engine, which in turn presents links to the matching resources in the repository, a document title, or possible summary information in the form of a short abstract of the original document. This abstract may be generated automatically and may contain the essence of the document. The user must then determine the relevance or importance of a document by reviewing the title and/or the abstract of the document presented in the result page of the search.

The larger the result set, the longer it takes the user to review the document titles and abstracts of the search results. Research has shown that a typical user will only carefully review the first five to ten result summaries for a particular search. However, search results may contain several hundred or several thousands of hits. Techniques, such as Boolean query language, may be used to limit and narrow down the number of hits.

A result set of ten to twenty hits may still take considerable time and effort to review because of the time required for reading the title and abstract. To really ensure whether a document is an ideal match to the search query, a user still has to open (i.e., view) a document. This means, however, that by clicking on a hyperlink (URL) and accessing a document resource with a web browser, the document content must be downloaded from the server to the client before viewing. It may take a considerable amount of time to access the document which therefore slows down the whole process. After downloading a document from the server, the user may then determine that the downloaded document is not a good match for the original search query. The user may then continue to read through the rest of the original result page and skim other abstracts looking for a more promising document. As a result of this process, a user typically has to download several documents until there is a good match for the original search query.

Documents with large amounts of text data may be rendered and then resized in order to create a visual abstract (also known as a thumbnail). As is well known to one skilled in the art, rendering means to process a document for representation. For example, an HTML document includes data and format instructions (i.e., tags). The format instructions need to be rendered before it can be displayed in its intended way. Rendering is typically done with a web browser such as Netscape Navigator or MS Internet Explorer. The rendering engine of the web browser essentially processes format instructions and converts them into graphical elements, determines the layout and calculates the overall appearance of the document.

However, after rendering and resizing the original body of text of the abstracts may not be readable because the font is too small. Moreover, with today's standard screen resolution, it may not be possible to produce a readable font in this size. It would be helpful for the user to read the headings or title and be able to determine whether a document is desirable for further reading. However, resizing algorithms use proportional resizing. The body text, which cannot be displayed at this size, will be reduced to the same size as the heading. It would be helpful to resize the body text and use this additional space to enlarge the headings and titles so that the user can read them.

SUMMARY OF THE INVENTION

In view of the foregoing and other problems of the conventional methods, it is, therefore, an object of the

US 7,177,948 B1

3

present invention to provide a method and apparatus of efficiently processing search results obtained in response to a user query.

The method according to the present invention may include examining document pointers returned by a search engine to identify a source from which the documents are available, obtaining the documents from the source and generating at least two visual abstracts for a desired document. Each of the visual abstracts is of a different size. A stream of data may be formatted such that when the data is displayed, a smaller one of the visual abstracts appears adjacent to a corresponding search result. A larger visual abstract may be displayed on the display screen when a cursor is moved over the smaller one of the visual abstracts. The larger visual abstract may also be removed from the display screen when the cursor is moved away from the smaller visual abstract.

The method according to the present invention may also include processing search results obtained in response to a user query. Document pointers returned by a search engine may be examined to identify a source from which the documents are available. The documents may be obtained from the source. Visual abstracts may be generated for each of the documents. Each visual abstract may be formed by manipulating a corresponding source document so as to enhance the visibility of at least a first portion of the source document while degrading visibility of at least a second portion of the source document. A stream of data may be formatted such that when the data is displayed on a display screen, each visual abstract appears adjacent to a corresponding search result.

It is also an object of the present invention to provide a computer system for searching for a document. This system may include a client system and a server system. The client system may be capable of supplying a search request to the server system. The server system may provide abstracts of documents to the client system. Further, the client system may display the abstracts on a display screen. The abstracts may include a written abstract and a first visual abstract of the documents. The server system may create a second visual abstract of one of the documents. The second visual abstract may be larger than the first visual abstract and the client system may display the second visual abstract when requested by a user.

Other objects, advantages and salient features of the invention will become apparent from the following detailed description taken in conjunction with the annexed drawings, which disclose preferred embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described in detail with reference to the following drawings in which like reference numerals refer to like elements and wherein:

FIG. 1 shows a result page following a typical online search;

FIG. 2 shows a medium-sized thumbnail of a selected document;

FIG. 3 shows the result page after the cursor has been moved from the result page;

FIG. 4 shows the distributed client and server system according to the present invention;

FIG. 5 shows a web page having a title and a body of text;

FIG. 6 shows a web page having an enhanced title;

FIG. 7 shows a web page having an enhanced title and proportioned resizing;

4

FIG. 8 shows another client and server system according to the present invention;

FIG. 9 is a schematic block diagram of the structure of an exemplary information handling/computer system for use with the present invention; and

FIG. 10 illustrates a medium for storing a program for implementing the method according to the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

The present invention aims to enhance the user's ability to perform online searching. This may be accomplished by creating (and caching) a medium-sized thumbnail of a document so as to better view the document. This may also be accomplished by enhancing the title or heading of a document or this may also be accomplished by dynamically presenting a visual abstract of a document.

A system according to the present invention may enhance searching techniques by displaying a medium sized visual abstract (i.e., image thumbnail) on demand. This provides the user with a clue about how the document looks as well as provides a preview of the selected document's content. A preferred method provides a medium sized thumbnail (around 300×200 pixels) in order to get a fairly good impression of the actual document. With this medium sized document, the user can read headlines or topics and recognize images and fonts much better than prior art methods. By reviewing the medium sized thumbnail, the user can determine whether a document is a good match and therefore needs to be downloaded. Accordingly, there is no need to download the entire document from the server to the client if the content of the medium size thumbnail does not properly match the original search query.

A medium sized thumbnail of a document can be obtained using existing image compression technologies in order to create a document requiring approximately 10–14 Kbytes as compared with an original document that requires more than ten times the amount of memory. The average size of a web document is between 30–100 Kbytes.

According to the preferred method, a medium sized thumbnail may be downloaded from the server to the client on demand. For instance, when a user moves a pointing device, such as a mouse cursor, over a summary abstract of a desired document listed in a result page, a medium sized thumbnail may be generated and displayed. The medium sized thumbnail is preferably only created for potential documents that appear to match a query. Generation of the medium sized thumbnail (also called medium sized visual abstract) is preferably done on the server side. The server preferably uses a caching mechanism to store the medium sized visual abstracts in a cache database so that users who later access the same document need not regenerate the medium sized abstract. The database may be programmed to store the medium sized visual abstract for a specific amount of time and then delete the abstract to conserve space. The client side of the system includes software that monitors the user's behavior, handles these events (such as downloading the medium sized thumbnail) and performs the associated actions (i.e., displaying the medium sized thumbnail).

FIG. 1 shows a typical result page 5 based on an online search. The result page 5 may contain hyperlinks 10 to external resources that matched the original query. The result page 5 generally includes a short summary description 12 and a visual abstract (i.e., thumbnail) image 14 for each

US 7,177,948 B1

5

document found in the search. The visual abstract image **14** is not necessary but is done merely for convenience.

The user skims the written abstracts **12** and visual abstract images **14** to determine the best match. In this example, the user believes that document number **3** is the best potential match for his query. However, from reading through the written abstract **12** and viewing the visual abstract image **14**, the user is not really sure whether document number **3** is really a good match. Rather than clicking on the third hyperlink **10** to download the entire document from the server to the client, which can take a considerable amount of time, the user may move the mouse pointer (or other pointing device or cursor) over the visual abstract image **14**.

When the mouse pointer (or cursor) is moved over one of the visual abstract images **14**, a medium sized visual abstract (i.e., thumbnail) **16** is requested from the server and will ultimately be displayed on the client side as shown in FIG. **2**. The user will better recognize the headline, logos, images and title on the medium sized thumbnail **16** as compared with the smaller visual abstract image **14**. Using this additional information, the user can better determine whether the document is worth downloading, or the user may decide to continue to skim the remaining summaries to find a better document.

After a user has reviewed the medium sized visual abstract **16**, he/she may move the mouse pointer to another location on the result page. The system is preferably programmed such that when the mouse pointer leaves the region of the smaller visual abstract image **14**, then the medium sized thumbnail **16** disappears from the screen. FIG. **3** shows the result page **5** after the medium sized thumbnail **16** has been removed from the screen. In order to speed up future processing, the medium sized thumbnail **16** is cached in a database on the client side. Therefore, if the user decides to take a second look at the medium sized thumbnail **16** of the third document, then the image will be directly loaded from the cache database rather than having to regenerate the medium sized thumbnail **16**.

The system is preferably implemented as a distributed client-server application as described below with respect to FIG. **4**. This disclosed system is not limiting as other systems that perform the above disclosed method are also within the scope of the present invention. As one skilled in the art would understand, the system preferably comprises software components.

On the client-side of the system, the event handler **20** tracks actions of the user. Typically users use pointing devices, such as a mouse, to scroll and move through displayed results. These movements are evaluated by the event handler **20**. If a user moves the mouse pointer over a specific spot on a result item, or preferably over a (small) visual abstract **14**, the event handler **20** triggers an event to the image requester **22** that contains the result item number/id (e.g., document number) and the URL of the requested document. The event handler **20** may also be responsible for hiding or discarding the medium sized visual abstract **16** on the client side once it is no longer needed.

The image requester **22** requests the medium sized thumbnail **16** of a document from the server. The request may be served either from a local cache on the client side via a HTTP request to the server side. The image requester **22** obtains the medium sized thumbnail **16** and passes it to the web browser for display on a display screen. If an error occurs (e.g. a medium sized thumbnail **16** cannot be loaded or generated), then the user will not be able to view the medium sized thumbnail **16**.

6

The above-described client-side components and their basic functionalities are already integrated into most modern web browser technologies. These web browsers provide an application programming interface (API) for scripting languages to achieve the functionalities discussed above.

The server-side components interact closely together to achieve the desired result. The session manager **24** identifies user sessions and retrieves corresponding user settings. For example, the user may turn the medium sized visual abstract feature off. This is desirable if a user uses a text-based web browser (e.g. Lynx) where he/she is not able to view images. The session manager **24** forwards the request to the URL loader **26**.

The URL loader **26** looks to the local cache (i.e., cache database **30**) by asking the cache manager **28** whether a medium sized thumbnail **16** for the requested document is already stored in the cache database **30**. This saves time and increases the overall performance of the system. The system may also include additional component(s) that detect idle cycles of the system and then uses these to generate the medium sized thumbnail **16** in advance.

When a medium sized thumbnail **16** cannot be retrieved from the cache database **30**, the URL loader **26** downloads the requested document from the repository (such as the Internet or database) and passes the contents to the rendering engine **32**.

The rendering engine **32** renders the document after receiving the document from the URL loader **26**. After a successful rendering of the document, the result is passed to the image processor **34**.

The image processor **34** captures the rendered document and resizes the image so that it will have a "medium" size in accordance with the present invention. A "medium" size preferably has a width between approximately 250 to 400 pixel units, and a height of approximately 170 to 300 pixel units. JPEG compression results in an image size of between 10 Kbyte to 18 Kbyte on the average. After the image is created, the image processor **34** stores the image in the cache manager **28**. The image is passed to the image requester **22** on the client side so that the medium sized thumbnail **16** can be displayed.

As discussed above, the medium sized thumbnails **16** are very useful to enable users to skim faster through a large result set. A medium sized thumbnail **16** is preferably created for only one document at any time so as to save time and network bandwidth.

Another feature of the present invention relates to enhancing the visual abstract (i.e., the thumbnail image) by enlarging the title, headings and logo of a document in the thumbnail image. It is helpful to the user to be able to read the title and headings of the text as the body of the text is generally not readable until the document is fully downloaded.

To illustrate the problem, FIG. **5** shows a typical web page document containing a title (or heading) and a body of text **42** in small text. When generating the visual abstract, the present invention may detect portions of the document that are more important (e.g. heading, title) and extract them into an intermediary temporary document. The temporary document may then be passed to a rendering engine and then resized. The result is a small thumbnail in which the user is able to read the heading and/or the title.

For the example shown in FIG. **5**, the system preferably extracts the heading "Culture and Technological Obsolescence". The detection is done by parsing the original document to create a new temporary document that enhances important information. Parsing of the document is prefer-

US 7,177,948 B1

7

ably done using a standard HTML parser. The parsing process looks for headlines that are marked in HTML using tags such as the <TITLE> tag. These structure tags typically contain relevant information about the topic of a document. FIG. 6 shows an example of the temporary document in which the title/heading has been enhanced to create an enhanced title/heading 44. In the temporary document, the body of text 42 preferably has not been resized.

After the rendering, standard proportional resizing can be applied to the temporary document to create the visual abstract as shown in FIG. 7, which includes a small visual abstract 50 with enhanced title/headings 44.

The enhanced title/heading 44 allows the user to more easily read the heading. However, the body text may still not be readable. The present invention may enhance the visual abstract so that the user will be able to determine whether it is worth taking a closer look at this document. This has several advantages. First, users will be able to identify headings and/or a title of a small visual thumbnail. Second, the invention will save downloading time and network bandwidth because after looking at this small thumbnail, the user may skip downloading the original document. Third, the invention greatly improves the overall value of a search results page because images are easier to skim than text documents.

Another feature of the present invention is to dynamically present a visual abstract of a result item (document) of a search result set. This takes advantage of the ability to recognize images faster than written text. Thus, a result page will present a list of images containing a visual abstract of the original document along with links/pointers to the resource, title and text abstract as additional/optional items.

The visual abstracts may be dynamically created "on the fly" as opposed to a static approach, which has a disadvantage that it cannot quickly reflect changes of a web document. By using a static approach, the visual abstract is not synchronized within a short period of time. The user may then review an incorrect visual abstract representation of the document. Moreover, the present invention preferably generates visual abstracts only if requested. The system generally will not generate abstracts for documents that have not been requested.

Because rendering and capturing of a larger result set can take a considerable amount of time, the system provides a caching mechanism to enhance the overall performance.

This system according to the present invention preferably works together with a text based search engine. The user submits a query to the search engine. The system analyzes the search results and generates a visual abstract of the original document. Then, the rendered document is converted to an image format (JPEG, TIFF) and the image is resized to a smaller size (i.e., a thumbnail size). The rendering and image converting process is a time consuming task, which can be done off-line for performance reasons. As a result, the modified result page of the search engine contains visual abstracts (thumbnails) of the documents rather than text based summaries.

The system will now be described that performs all the tasks of generating a visual abstract during a user query process on the fly. For performance reasons, the complete process can be enhanced using existing caching technologies, which is handled by the cache manager 28 as shown in FIG. 8.

When a user issues a query 60, the session manager 24 receives the request. The session manager 24 tracks user sessions using existing web technologies (e.g. Cookies, Active Server Pages) as well as analyzes the user settings to

8

determine display preferences. Users may enable or disable the visual abstracts. If the visual abstracts are disabled, the system passes the user query to the search engine system, waits for the returned results and forwards the returned results to the representation manager 62.

If the visual abstracts are enabled, it passes the user query to the search engine system, waits for the returned results and forwards the returned results along with session information to the URL loader 26.

The URL loader 26 takes a list of URLs as an input and then loads the document associated with an URL. When a document is loaded, it forwards the document along with the session id to the rendering engine 32. If a document cannot be loaded, an error message may be passed directly to the representation manager 62 so that the representation manager 62 can skip this entry.

For performance reasons, the URL loader 26 asks the cache manager 28 whether the desired URL was previously loaded. In this case it can directly retrieve the rendered and captured image from the cache manager 28 and pass the visual abstract to the representation manager 62. This saves a lot of work and time and therefore speeds up response time.

The rendering engine 32 takes a HTML document as an input and renders the document. This rendering process can be compared with viewing a HTML document within a web browser. The web browser parses the document and generates the visual representation. However, the result of the rendering process may not be immediately presented to the user. It's an intermediary result that will be passed to the image capturer 64. If the rendering process fails, an error message will be passed to the representation manager 62 so that the representation manager 62 can skip this entry.

The image capturer 64 takes a screen capture of the rendered document and generates an image thumbnail by resizing the original image. This image may then be passed to the cache manager 28 along with a time stamp for later reuse. This prevents the system from skipping rendering and image processing for documents that were already rendered. The image thumbnail along with session information and URL is passed to the representation manager 62, which will construct the result page for the user and integrate the visual abstracts to the summary abstract listing.

Finally, the cache manager 28 stores image thumbnails (i.e., visual abstracts) in a cache database 30 and keeps track of the rendered documents along with a time stamp for each resource. Before the time intensive process of rendering and image processing is initiated, the system first queries the cache manager 28 to determine whether the document is already processed. If so, then the cache manager 28 simply returns the visual abstract.

In summary, the system dynamically creates visual abstracts for search results. The system focuses on the dynamical visual abstract (thumbnail) generation for documents obtained as a search result rather than presenting visual thumbnails of static content. Further, the system does not use the visual abstract (thumbnail of a document) for querying but rather uses the visual abstract to help users identify important contents faster by looking at an image thumbnail.

While the overall methodology of the invention described above generally relates to a client-server environment, the invention can be embodied in any number of different types of systems and executed in any number of different ways, as would be known by one ordinarily skilled in the art.

For example, as shown in FIG. 9, a typical hardware configuration 800 of an information handling/computer sys-

US 7,177,948 B1

9

tem incorporates the client side environment. The system preferably has at least one processor or central processing unit (CPU) **811**. The CPUs **811** are interconnected via a system bus **812** to a random access memory (RAM) **814**, read-only memory (ROM) **816**, input/output (I/O) adaptor **818** (for connecting peripheral devices such as disk units **821** and tape drives **840** to the bus **812**), user interface adapter **822** (for connecting a keyboard **824**, mouse **826**, speaker **828**, microphone **832**, and/or other user interface device to the bus **812**), communication adapter **834** for connecting an information handling system to the Internet, Intranet, a data processing network, etc., and a display adapter **836** (for connecting the bus **812** to a display device **838**). Additionally, an external controller **850** can be coupled to the system through the network and communications adapter **834**.

Further, while the present invention has been described primarily in terms of software or software/hardware configuration, the same or similar functions could be implemented in a dedicated hardware arrangement.

In addition to the hardware/software environment described above, a different aspect of the invention includes a computer-implemented method for searching for documents. As an example, this method may be implemented in the particular environment discussed above.

Such a method may be implemented, for example, by operating a computer, as embodied by a digital data processing apparatus, to execute a sequence of machine-readable instructions. These instructions may reside in various types of signal-bearing media.

Thus, this aspect of the present invention is directed to a programmed product, including signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital data processor to perform a method of searching for documents.

This signal-bearing media may include, for example, a random access memory (RAM) such as, for example, a fast-access storage contained within the computer. Alternatively, the instructions may be contained in another signal-bearing media, such as a magnetic storage diskette **900** shown exemplarily in FIG. **10**, directly or indirectly accessible by the computer.

Whether contained in the diskette, the computer, or elsewhere, the instructions may be stored on a variety of machine-readable data storage media, such as DASD storage (e.g. a conventional "hard drive" or a RAID array), magnetic tape, electronic read-only memory (e.g. ROM, EPROM, or EEPROM), an optical storage device (e.g. CD-ROM, WORM, DVD, digital optical tape, etc.), paper "punch" cards, or other suitable signal-bearing media including transmission media such as digital and analog and communication links and wireless. In an illustrative embodiment of the invention, the machine-readable instructions may comprise software object code, compiled from a suitable language.

While the invention has been described with reference to specific embodiments, the description of the specific embodiments is illustrative only and is not to be considered as limiting the scope of the invention. Various other modifications and changes may occur to those skilled in the art without departing from the spirit and scope of the invention.

What is claimed is:

1. A method of processing search results obtained in response to a user query, the method comprising:
providing document pointers returned by a search engine to identify a source from which documents are available, each said document pointer including a Uniform

10

Resource Locator (URL) displayed as part of a search result by said search engine;

generating at least two visual abstracts for at least one of said documents, each of said two visual abstracts being a thumbnail image of a different size, said visual abstracts being generated after first manipulating said document so as to enhance a visibility of at least a portion of said document, said manipulating being performed by filtering said document; and
formatting a stream of data such that when said data is displayed on a display screen regarding said at least one of said documents, a smaller one of said visual abstracts appears adjacent to a corresponding search result.

2. The method of claim 1, wherein said filtering is performed on an image in said document.

3. The method of claim 1, further comprising:

displaying a larger one of said visual abstracts on said display screen when requested by said user.

4. The method of claim 3, further comprising:

storing data relating to said larger one of said visual abstracts.

5. The method of claim 3, wherein said larger one of said visual abstracts is displayed on said display screen when a cursor is moved over said smaller one of said visual abstracts.

6. The method of claim 5, further comprising:

removing said larger one of said visual abstracts from said display screen.

7. The method of claim 6, wherein said larger one of said visual abstracts is removed from said display screen when said cursor is moved away from said smaller one of said visual abstracts.

8. A method of processing search results obtained in response to a user query, the method comprising:

examining document pointers returned by a search engine to identify a source from which documents are available, each said document pointer including a Uniform Resource Locator (URL);

obtaining said documents from said source;

generating a visual abstract for each of said documents, each visual abstract being a thumbnail image, each said thumbnail image comprising a visual similarity of said document as reduced in size, a title of said document ensured to be readable on each said thumbnail image;

formatting a stream of data such that when said data is displayed on a display screen, each visual abstract appears adjacent to a corresponding search result;

creating a larger visual abstract of at least one of said documents;

displaying said larger one of said visual abstracts on said display screen on demand; and

removing said larger one of said visual abstracts from said display screen.

9. The method of claim 8, further comprising:

storing data relating to said larger one of said visual abstracts.

10. The method of claim 8, wherein said larger one of said visual abstracts is displayed on said display screen when a cursor is moved over said smaller one of said visual abstracts.

11. The method of claim 8, wherein said larger one of said visual abstracts is removed from said display screen when said cursor is moved away from said smaller one of said visual abstracts.

US 7,177,948 B1

11

12. A method of processing search results obtained in response to a user query, the method comprising:
 examining document pointers returned by a search engine to identify sources from which documents are available, each said document pointer including a Uniform Resource Locator (URL);
 obtaining said documents from said sources;
 generating a visual abstract for each of said documents, each visual abstract being a thumbnail image, each said thumbnail image comprising a visual similarity of said document as reduced in size, a title of said document ensured to be readable on each said thumbnail image; formatting a stream of data such that when said data is displayed on a display screen, each visual abstract appears adjacent to a corresponding search result;
 determining whether a portion of a source document should be enhanced for visibility relative to another portion; and
 manipulating said source document determined to have a portion to be enhanced so that one portion therein is manipulated to improve a visibility while another portion therein is manipulated to degrade a visibility, wherein said manipulating includes filtering said source document.

13. The method of claim 12, wherein said filtering is performed on an image in said source document.

14. The method of claim 12, wherein said portion of said source document to be enhanced corresponds to at least one of a title and a heading of said source document.

15. The method of claim 14, wherein one of said title and said heading is enlarged as compared with a second portion of said source document.

16. The method of claim 15, wherein said second portion of said source document corresponds to a body of text of said source document.

17. A method of searching for a document, said method comprising:
 supplying a search request;
 providing abstracts of documents on a screen display that correspond to said search request, said abstracts including a written abstract that contains a summary of a contents of said documents and a first visual abstract of each of said documents;
 creating a second visual abstract of one of said documents, each of said first visual abstract and said second visual abstract respectively being a thumbnail image of said document, wherein said second visual abstract is larger than said first visual abstract;
 displaying said second visual abstract when requested by a user, said second visual abstract being displayed on said display screen when said user moves a pointing device over a corresponding one of said first visual abstract; and
 removing said second visual abstract from said display screen when said user moves said pointing device away from said corresponding one of said first visual abstracts.

18. The method of claim 17, wherein said first visual abstract is created after manipulating a source document so as to enhance visibility of at least a first portion of said source document.

19. The method of claim 18, wherein said first portion corresponds to at least one of a title and a heading of said document.

12

20. The method of claim 17, further comprising:
 storing data relating to said second visual abstract in a cache database.

21. The method of claim 20, further comprising:
 deleting said data relating to said second visual abstract in said cache database after a predetermined amount of time.

22. A program storage device readable by machine, tangibly embodying a program of instructions executable by said machine to perform method steps for processing search results obtained in response to a user query, said method comprising:
 supplying a search request;
 providing abstracts of documents on a screen display that correspond to said search request, said abstracts including a written abstract containing a summary of a contents of said documents and a first visual abstract of said documents;
 creating a second visual abstract of one of said documents, each of said first visual abstract and said second visual abstract respectively being a thumbnail image of said document, wherein said second visual abstract is larger than said first visual abstract;
 displaying said second visual abstract when requested by a user, said second visual abstract being displayed on said display screen when said user moves a pointing device over a corresponding one of said first visual abstracts; and
 removing said second visual abstract from said display screen when said user moves said pointing device away from said corresponding one of said first visual abstracts.

23. A computer system for searching for a document, said system comprising:
 a client system; and
 a server system,
 said client system supplying a search request to said server system, said server system providing abstracts of documents to said client system, said abstracts corresponding to said search request, said client system displaying said abstracts on a screen display, said abstracts including a written abstract of a contents of said documents and a first visual abstract of each of said documents, said server system creating a second visual abstract of one of said documents, each of said first visual abstract and said second visual abstract respectively being a thumbnail image of said document, said second visual abstract being larger than said first visual abstract when displayed on said screen display, said client system displaying said second visual abstract when requested by a user, said second visual abstract being displayed on said screen display when said user moves a pointing device over said first visual abstract, said second visual abstract being removed from said screen display when said user moves said pointing device away from said first visual abstract of said document.

24. The system of claim 23, wherein said first visual abstract is created after manipulating a source document so as to enhance visibility of at least a first portion of said source document.

* * * * *

EXHIBIT D



US006496206B1

(12) **United States Patent**
Mernyk et al.

(10) **Patent No.: US 6,496,206 B1**(45) **Date of Patent: *Dec. 17, 2002**

(54) **DISPLAYING THUMBNAIL IMAGES OF
DOCUMENT PAGES IN AN ELECTRONIC
FOLDER**

(75) Inventors: **Paul A. Mernyk**, Palo Alto, CA (US);
Steven Martin, Topsfield, MA (US);
Bevra S. Prasad, Milpitas, CA (US);
David L. Salgado, Victor, NY (US)

(73) Assignee: **Scansoft, Inc.**, Peabody, MA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/106,154**

(22) Filed: **Jun. 29, 1998**

(51) **Int. Cl.⁷** **G06F 3/00**

(52) **U.S. Cl.** **345/835; 345/764**

(58) **Field of Search** 345/333, 334,
345/335, 337, 339, 347, 338, 341, 353,
354, 369, 762, 763, 744, 707, 764, 808,
835, 839; 707/3, 4, 840, 314

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,648,249 A	3/1972	Goldsberry
3,705,956 A	12/1972	Dertouzos
4,177,354 A	12/1979	Mathews
4,387,395 A	6/1983	Shaphorst
4,424,575 A	1/1984	Clarke et al.

4,430,526 A	2/1984	Brown
4,475,239 A	10/1984	Raamsdonk
4,488,000 A	12/1984	Glenn
4,514,818 A	4/1985	Walker
4,516,156 A	5/1985	Fabris et al.
4,528,693 A	7/1985	Pearson et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	159400	10/1985
----	--------	---------

OTHER PUBLICATIONS

Macworld, "A Pair of Digitizing Tablets", published Mar. 1987 by PCW Communications, Inc., San Francisco, Calif. pp. 143-144.

"Method for Creating Annotation Data", IBM Technical Disclosure Bulletin, vol. 28, No. 4, Sep. 1985, pp. 1623-1628.

(List continued on next page.)

Primary Examiner—Raymond J. Bayerl

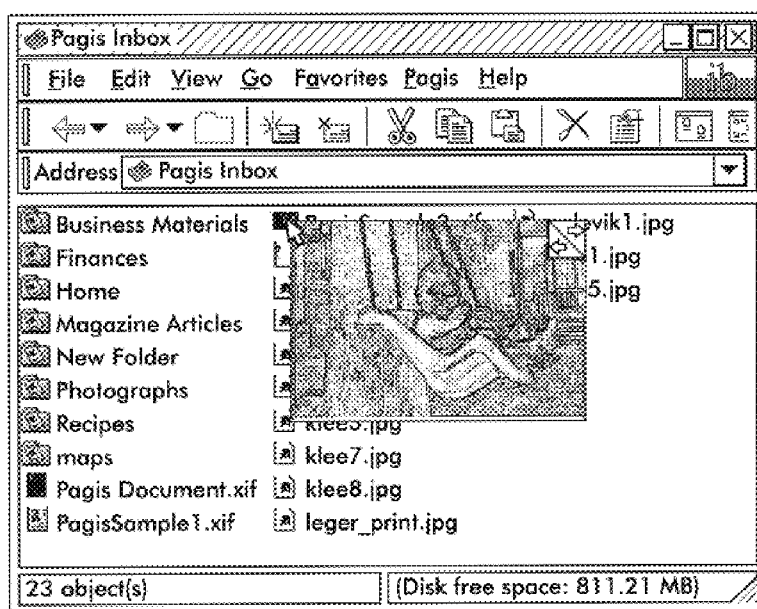
Assistant Examiner—Cao H. Nguyen

(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

In a graphical user interface for accessing a large number of files, such as text files, graphics files, or spreadsheets, a system allows quick glances of "thumbnails" or highly reduced versions of the files. When a folder is opened, every file in the folder is opened as a background operation and thumbnail data, such as a reduced image or text summary of the file, is derived and retained in a cache folder for quick access. When a cursor is touched, without a mouse-click, to a particular icon in the opened folder, the thumbnail for the file identified by the icon is accessed from the cache folder and displayed.

19 Claims, 5 Drawing Sheets



US 6,496,206 B1

Page 2

U.S. PATENT DOCUMENTS

4,528,988 A 7/1985 Wong
 4,552,991 A 11/1985 Hulls
 4,562,304 A 12/1985 Ward et al.
 4,570,033 A 2/1986 Hulls
 4,575,580 A 3/1986 Jandrell
 4,577,057 A 3/1986 Blesser
 4,580,007 A 4/1986 Searby
 4,582,955 A 4/1986 Blesser
 4,587,633 A 5/1986 Wang et al.
 4,616,336 A 10/1986 Robertson et al.
 4,633,416 A 12/1986 Walker
 4,633,436 A 12/1986 Flurry
 4,638,119 A 1/1987 Blesser et al.
 4,644,102 A 2/1987 Blesser et al.
 4,645,238 A 2/1987 Vincent et al.
 4,649,380 A 3/1987 Penna
 4,675,665 A 6/1987 Halliwell
 4,677,428 A 6/1987 Bartholow
 4,688,031 A 8/1987 Haggerty
 4,714,918 A 12/1987 Barker et al.
 4,723,836 A 2/1988 Kono et al.
 4,734,619 A 3/1988 Havel et al.
 4,742,473 A 5/1988 Shugar
 4,755,809 A 7/1988 Ikegami et al.
 4,785,564 A 11/1988 Gurtler
 4,899,136 A 2/1990 Beard
 4,926,484 A 5/1990 Nakano
 5,008,853 A 4/1991 Bly
 5,060,135 A 10/1991 Levine et al.
 5,072,412 A 12/1991 Henderson, Jr. et al.
 5,161,213 A 11/1992 Knowlton
 5,355,447 A 10/1994 Knowlton
 5,436,637 A 7/1995 Gayraud et al.
 5,499,108 A 3/1996 Cotte et al.
 5,500,935 A 3/1996 Moran et al.
 5,500,936 A 3/1996 Allen et al.
 5,517,332 A 5/1996 Barry et al.
 5,625,833 A 4/1997 Levine et al.
 5,644,692 A 7/1997 Eick
 5,715,416 A * 2/1998 Baker 345/349
 5,761,655 A * 6/1998 Hoffman 707/4

5,819,261 A * 10/1998 Takahashi et al. 707/3
 5,995,101 A * 11/1999 Clark et al. 345/338

OTHER PUBLICATIONS

“The Smalltalk Environment”, by Tesler, L., BYTE Magazine, Aug. 1981, pp. 90–144.
 “The Lisa Computer System: Apple Designs a New Kind of Machine”, by Williams, G., BYTE Magazine, Feb. 1983, pp. 33–50.
 “The Apple Macintosh Computer: Mouse–Window–Desktop Technology Arrives for Under \$2500”, by Williams, G., BYTE Magazine, Feb. 1984, pp. 30–54.
 “System for Integrating and Collating Audio and Text, and for Text Creation and Editing”, by P.D. Welch, IBM Technical Disclosure Bulletin, vol. 16, No. 2, Jul. 1973, pp. 500–503.
 “Put—That—There”: Voice and Gesture at the Graphics Interface by Richard A. Bolt, in ACM 1982, pp. 262–270.
 Goodman, Danny, The Complete Hyercard Handbook (Bantam Books 1987), pp. 32–34.
 Microsoft Corporation, Show Partner User’s Guide (1986), Chaps. 3, 4, pp. 27–48.
 “Designing the Star user Interface,” by Smith, D.C., et al., BYTE Magazine, Apr. 1982, pp. 242–282.
 “Visual Languages: A Tutorial and Survey,” by Chang, Shi–Kuo, I.E.E.E. Software, Jan. 1987, v.4, No. 1, pp. 29–39.
 Flow Chart Generator, Patent Abstracts of Japan, vol. 7, No. 198 (P–220) (1343) Sep. 2, 1983, JP–A–58–96360 (Abstract Only).
 IBM Technical Disclosure Bulletin, Include Non–Text Objects In Margin Text, vol. 30, No. 7, Dec. 1987, p. 348.
 Stefik et al., “WYSIWIS Revised: Early Experiences with Multiuser Interfaces”, ACM Transactions on Office Information Systems, vol. 5, No. 2, Apr. 1987, pp. 147–167.
 Pratt, William, “Digital Image Processing”, A Wiley–Interscience Publication, 1978, pp. 322–323.

* cited by examiner

U.S. Patent

Dec. 17, 2002

Sheet 1 of 5

US 6,496,206 B1

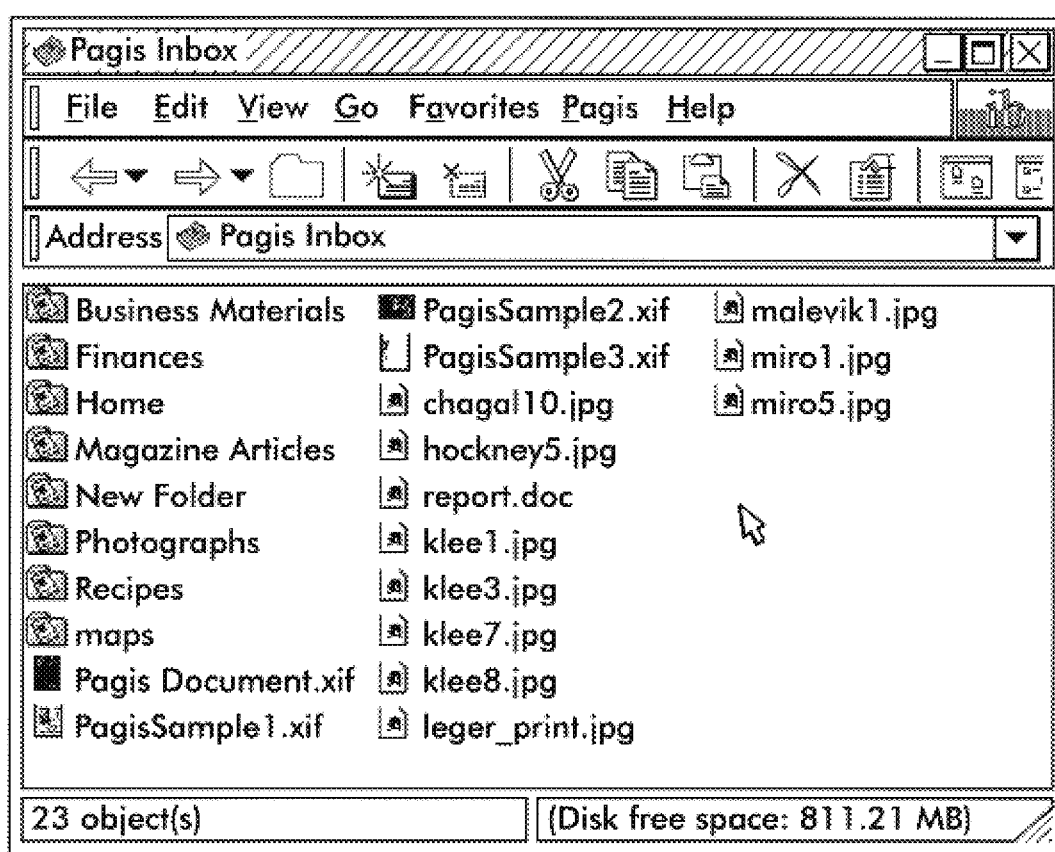


FIG. 1

U.S. Patent

Dec. 17, 2002

Sheet 2 of 5

US 6,496,206 B1

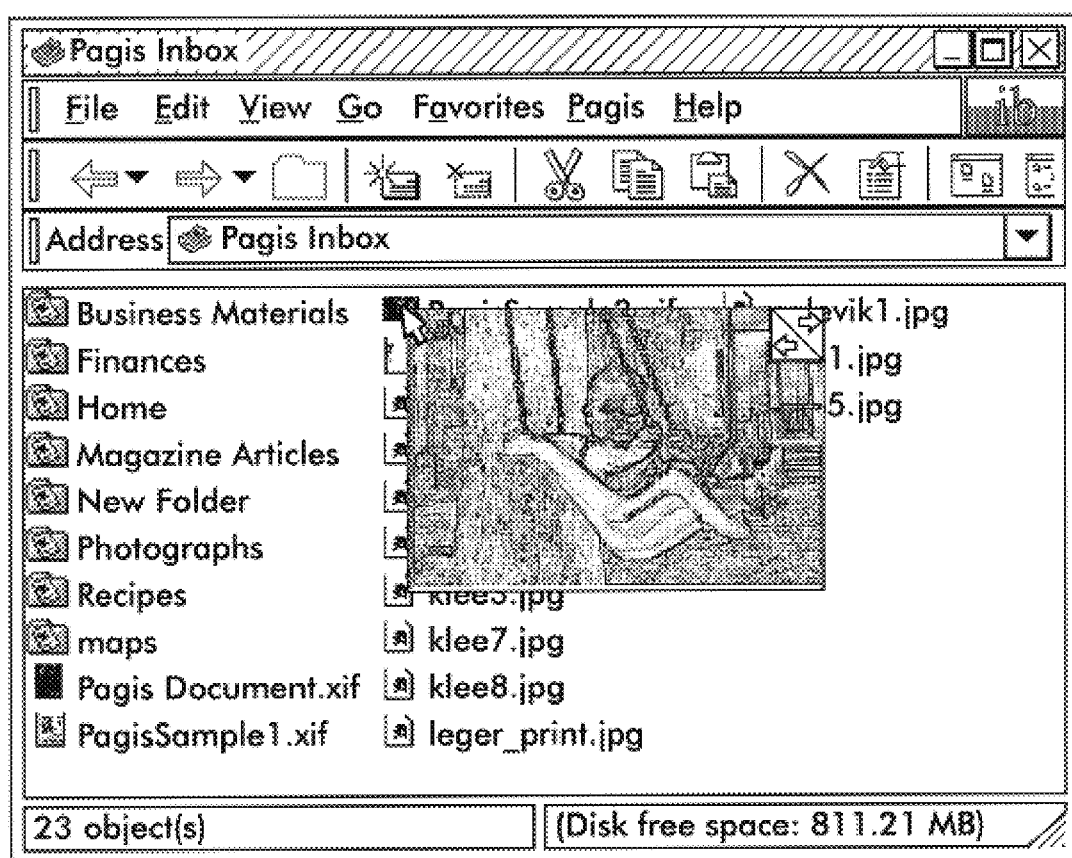


FIG. 2

U.S. Patent

Dec. 17, 2002

Sheet 3 of 5

US 6,496,206 B1

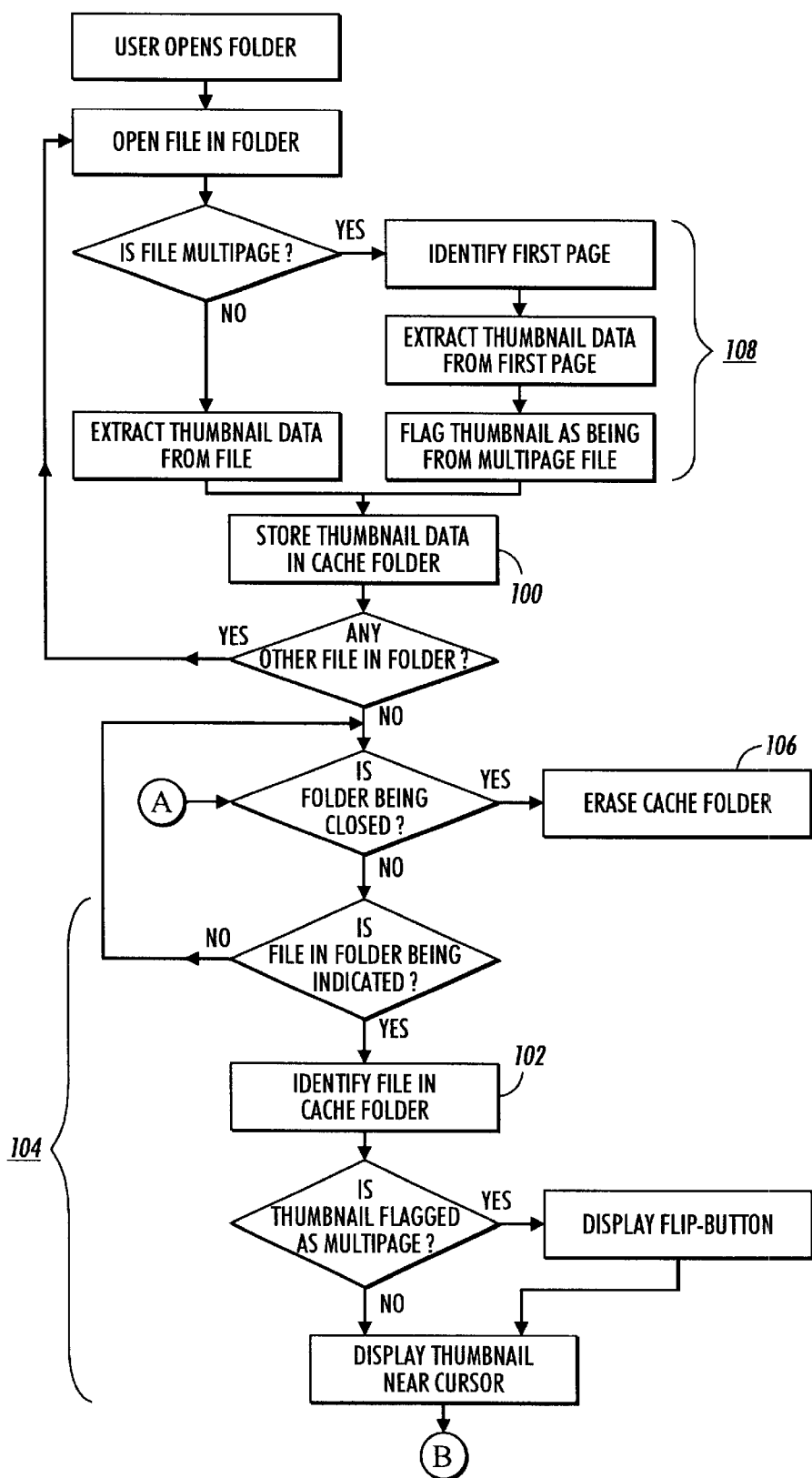
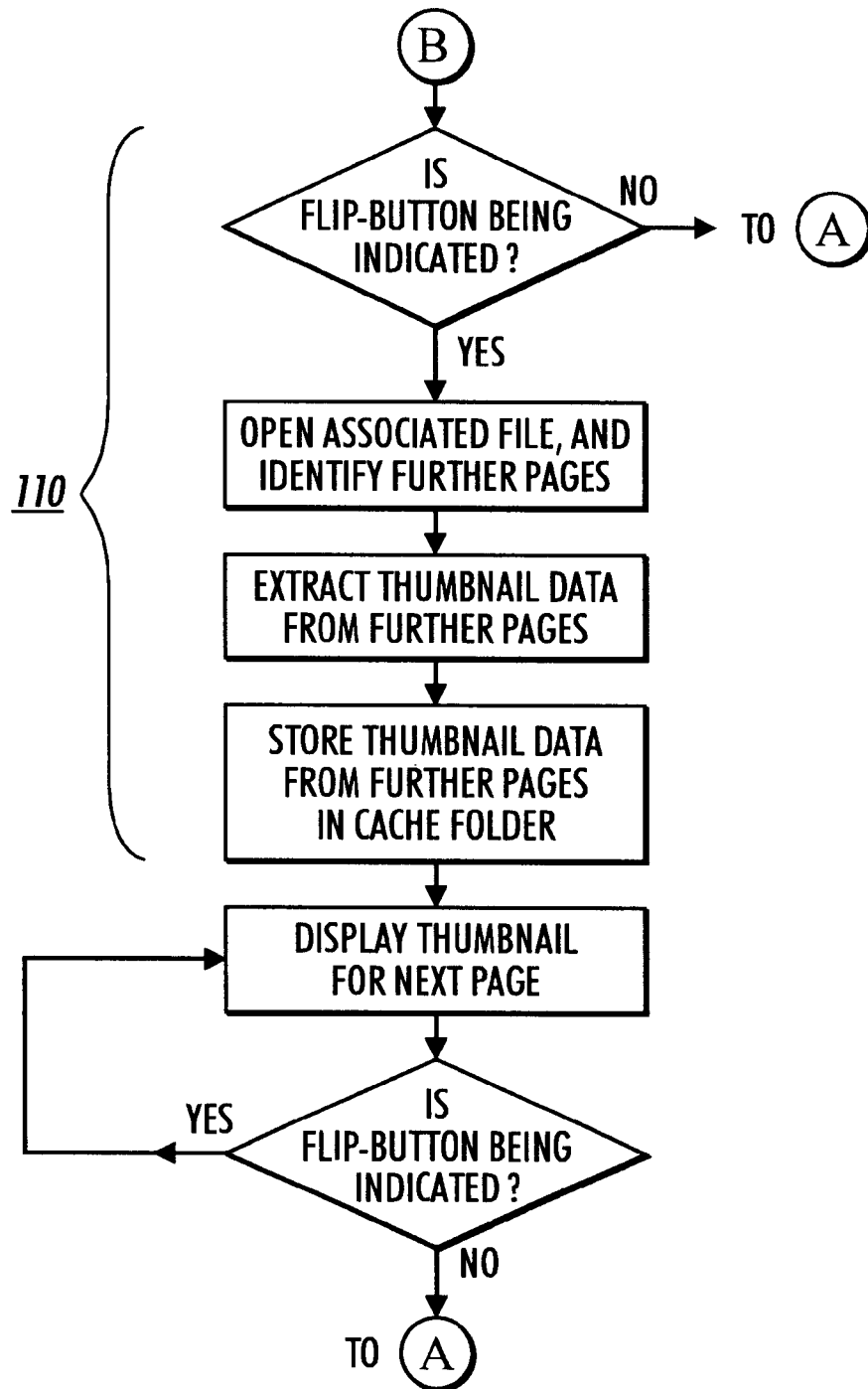


FIG. 3A

**FIG. 3B**

U.S. Patent

Dec. 17, 2002

Sheet 5 of 5

US 6,496,206 B1

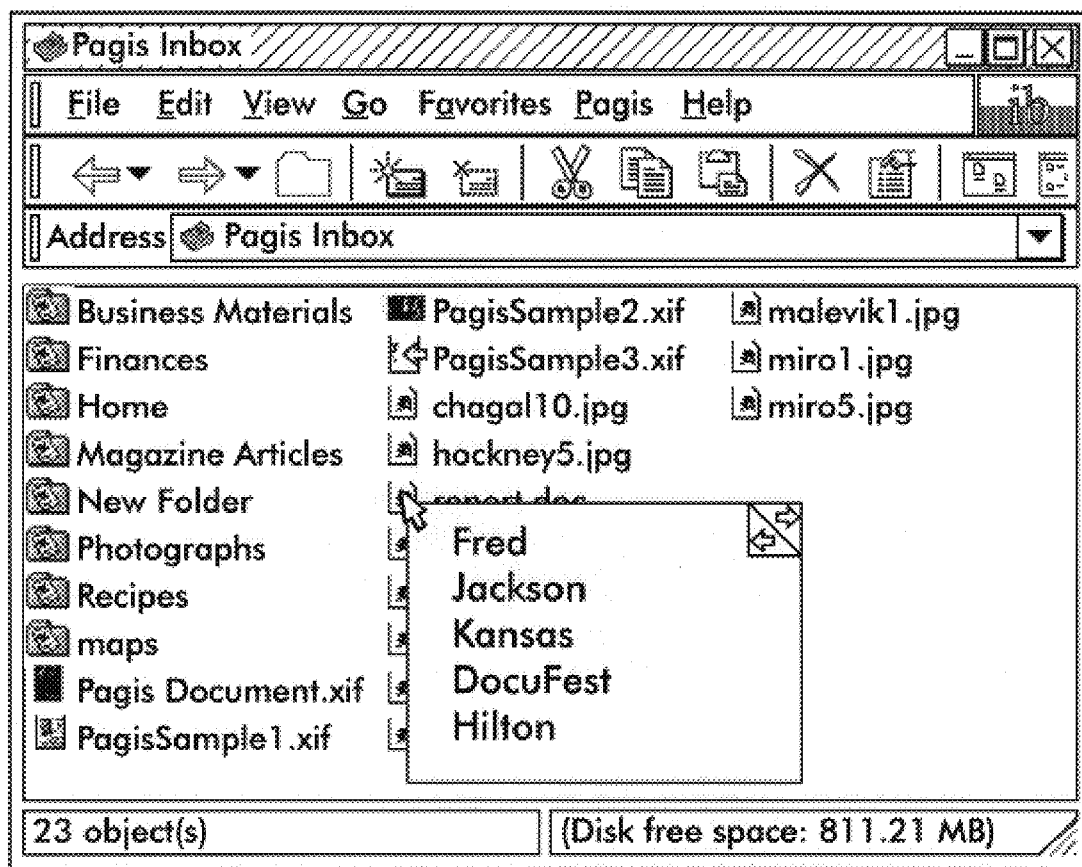


FIG. 4

US 6,496,206 B1

1

DISPLAYING THUMBNAIL IMAGES OF DOCUMENT PAGES IN AN ELECTRONIC FOLDER

FIELD OF THE INVENTION

The present invention relates to graphical user interfaces such as used with general-purpose computers, and relates more specifically to systems for obtaining overviews of icons or other display graphics representing relatively large quantities of data.

BACKGROUND OF THE INVENTION

In the computer industry, use of graphical user interfaces, or GUIs, is well known for enabling a user to select a particular file of data (such as a word-processing file, or a graphics file) from a large available selection. A GUI is a type of display format that enables a user to operate a computer by pointing to pictorial representations, such as "windows" and "icons," displayed on a screen device. A window is a rectangle displayed on the screen that affords a user work space within a program. In typical operation, the user may move the window about on the screen, change its size or shape, enlarge it to fill the screen, close it entirely, or change how much of its contents are displayed.

To navigate within a GUI, such as to select a particular file to be opened, most systems employ a screen cursor or pointer, typically displayed as a small arrow which allows the user to select individual points on the screen. In operation, the cursor is moved to a desired screen location in response to movements of a pointing device (e.g., a mouse, trackball, or equivalent) by the user. Besides effecting cursor movement, most pointing devices include one or more switches or "mouse buttons" for specifying additional user input or user events. Since many user choices may be entered through use of a pointing device instead of input with a keyboard, the need for the user to memorize special commands is lessened.

With particular reference to accessing data desired to be viewed, a standard arrangement of a GUI is to provide a hierarchy of containers into which individual files can be organized. For instance, a set of "files" can be placed in a "folder," a set of "folders" can be placed in a "drawer," and so forth, up to items which may be characterized as cabinets or drives. For present purposes, the important fact is that the basic unit of a set of data which may be wished to be viewed is here called the "file": a file may be a word-processing document, a graphics-program document, a spread sheet, or some other form of data that is capable of making sense standing alone. Also for present purposes, all containers which may contain one or more files will be called a "folder," even though in different contexts folders may exist within other folders, and higher-level metaphors may be used, such as drawer and cabinet.

When a folder is opened in the context of a GUI, there is typically displayed a rectangular space on the screen, and within this rectangular space is displayed a set of icons, each icon being associated with one file (such as a text or graphics file) in the folder. In common operating systems such as Macintosh® or Windows™, documents of a particular type, such as word-processing documents, are typically all assigned icons of an identical basic appearance, such as of a sheet with a folded corner, or a stylized capital W. By and large, the only way to distinguish between icons of a type relating to different files is to associate with each icon a short file name which is displayed underneath the icon. However,

2

very often, a file name which makes sense to the creator of the file will be largely meaningless to another user.

The present invention is a utility, which can be superimposed over an existing operating system such as Windows™, that enables a user to quickly identify the basic contents of each of a large number of files which are identified as icons within a folder in a GUI.

DESCRIPTION OF THE PRIOR ART

U.S. Pat. No. 5,436,637 describes a graphical user interface including providing "hints" for screen objects of interest. When a cursor merely touches a particular icon on the screen, a status frame or window, which is positioned in a non-intrusive fashion below or to one side of an active portion of the user interface, is continually updated with descriptive information relating to a particular icon as a screen cursor moves from one icon to another. The patent further contains a general background description of a utility known as "balloon help," in which graphical buttons a toolbar in a graphical user interface are provided with "pop-ups" or "balloons" which appear next to the cursor to explain the function of the button.

U.S. Pat. No. 5,500,935 discloses a graphical user interface system which allows the user to implement pop-up menus and gestural marks when the input location in the GUI is restricted, such as in a corner region. In such a situation, the system translates the desired pop-up menu from one location where space is limited to another location where space is essentially unlimited, and then provides an indicator or a guide to direct the user to make the required movement of the cursor to the unrestricted location.

U.S. Pat. No. 5,500,936 discloses a multi-media slide presentation system which can be superimposed on a standard computer GUI. Each slide in the multimedia slide presentation may contain photographs, text, graphics, and charts. By actuating and releasing a control button of the mouse or trackball, a pop-up menu is displayed to aid the user to make selections that operate on the slides and objects on the slides. The text menu items may also indicate pop-up menus when selected.

U.S. Pat. No. 5,644,692 discloses a specialized GUI for displaying information about a very large number of entities, such as files. The entity representations in the GUI are contained in columns which represent contexts for the entities. In one embodiment, the entities are lines of text and the contexts are files containing the lines. A selector with a set of colored fields corresponding to values of attributes of the entities is provided. When either an entity representation or a selector field is activated by means of a pointing device, the selector field and all of the entity representations for entities of the attribute value corresponding to the selector are turned on and appear in the same color. The pointing device may also be used to specify an entity representation for detailed viewing.

SUMMARY OF THE INVENTION

According to one aspect of the present invention, there is provided a method of displaying a thumbnail relating to an electronically-stored file in an electronically-stored folder, the folder being capable of retaining a plurality of files. At least one icon relating to a file retained in the folder is displayed. Thumbnail data is derived for each file in the folder. The thumbnail data for each icon in the folder is stored in a cache, and an association is created between each file in the folder and the thumbnail data relating to the file in the cache. When the icon relating to a file in the folder is

US 6,496,206 B1

3

indicated by a cursor, the thumbnail data relating to the file relating to the icon is retrieved from the cache and displayed.

According to another aspect of the present invention, there is provided a method of displaying a thumbnail relating to an electronically-stored file in an electronically-stored folder, the folder being capable of retaining a plurality of files. At least one icon relating to a file retained in the folder is displayed. When the icon relating to a file is indicated by a cursor, the thumbnail data relating to the file related to the icon is displayed by touching the cursor to the icon without a mouse-click.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a representation of a typical folder display, as in the prior art;

FIG. 2 is a demonstration of the display of a thumbnail of a graphics file according to the present invention;

FIGS. 3A and 3B show a simplified flowchart illustrating the basic steps of the method of the present invention; and

FIG. 4 is a demonstration of the display of a text-intensive thumbnail according to one aspect of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

For purposes of elucidating the specification and claims, the following terms will be defined:

A "thumbnail" is a quantity of data which is derived from a larger quantity of data, such as a file. To obtain a thumbnail, one page image from the file can be opened and optically reduced or scaled to a smaller size; or, alternately, discrete portions of data from the file can be pulled from the file, such as key words, a pre-written or artificially derived summary of the data, a title, a list of column headings, any text strings which are found to be of larger than usually type size which would be consistent with headlines, etc. The "thumbnail data," as used in the claims, is a quantity of data, which may be in form example ASCII format, which is culled from a file and which may be displayed in another format in the resulting thumbnail.

A "folder," for purposes of the present description and claims, is a term which can be applied to any container that can hold one file or a plurality of files.

An "icon" is in a basic sense a bitmap of a predetermined design, which the human user can identify as relating to a file, or at least to a file of a certain type. In addition, within the scope of the invention, the term "icon" can also be applied to a displayed file name, particularly in the context of a displayed tree of folders having files therein.

A "cursor" is a displayable pointing device, which is typically in the form of an arrow bitmap which is used to point to a particular icon, although this term could be applied broadly to encompass other methods of pointing to an icon, such as using a touch-screen, electronic pen, or a special alphabet.

A "background operation" is an action performed by the operating system of a computer, in a manner which is generally not apparent to the user through the display, or at least is performed by the operating system in a manner whereby the user can perform some other function while the background operation is taking place.

FIG. 1 is an example of a graphic display of a "folder," in the specific case an Inbox having a toolbar and pull-down menus associated therewith, as would be used in conjunction with the present invention. As can be seen, the displayed

4

folder includes a rectangular area with icons of the files contained in the folder. It will also be noted that some of the items in the folder are themselves folders, and may contain further folders and files. The type of entity which appears in the folder, whether a subsidiary folder, a text file, or a graphics file, determines the appearance of the particular icon associated with the file name. For folders within folders forming a hierarchy, it is known in the art to display multiple layers of folders in a "tree" structure.

FIG. 2 is an example display of a "thumbnail" associated with a file in a folder, showing the desired result of the present invention. When the cursor, shown as the small arrow in FIG. 2, is merely touched onto an icon within the folder, that icon being associated with the file, a "thumbnail" of the image forming the file is displayed. As can be seen in FIG. 2, this "thumbnail," in the case of a graphics file such as shown as "PagisSample2.xif" a reduced version of the image is displayed in a manner which associates the thumbnail with the indicated icon. In this manner, a person having a folder with a large number of text or graphics files, each file having a cryptic file name, can quickly glance at a thumbnail derived from the file so that the user can identify the thumbnail with the file. (For illustrative purposes, what is shown in the present FIG. 2 is intended to represent a reduced-size photograph.).

To optimize consumer satisfaction, the thumbnails such as shown in FIG. 2 must appear essentially immediately after the cursor touches the icon in question. If too much time is taken in order to generate the thumbnail, this time could have just as well been used to open the particular file. The underlying method of the present invention thus contemplates using a "cache" memory, which is basically a file not directly apparent to the user, for obtaining just enough data to generate the thumbnails of the icons contained in the folder which is displayed to the user. By maintaining the thumbnail data in the cache memory, the thumbnail data for each individual icon being relatively small, the thumbnail for the various icons in the folder can be displayed to the user with a satisfactory speed after the particular icon is touched with the cursor. When a particular folder having files therein is opened by the user, a corresponding "cache folder" is created on an ad-hoc basis. The cache folder is a folder of files of thumbnail data, but each file in the cache folder corresponds to and is associated with a file in the folder which is displayed to the user.

FIGS. 3A and 3B show a simplified flowchart showing the basic operation of the method of the present invention. The functionality shown in the FIGS. can be embodied in code which is superimposed on the basic functions of an operating system, such as through the commercially-available utility "Shell Name Space Extension" in Microsoft® Windows 95™. Following the flowchart, the method of the present invention is activated when a user opens a folder containing files therein, causing the folder space to be displayed on the GUI with icons therein, each icon corresponding to a file such as shown in FIG. 1 above. Once the folder having the files is opened, every file in the folder is opened as a background operation and thumbnail data is derived therefrom, such as at step 100. In the case of a purely graphics file such as in .xif or .jpg or .tif format, this thumbnail data can be readily derived by sampling the graphical data to yield the relatively small thumbnail that will be displayed. In the case of a file, such as in a word-processing format, which is known to be text-exclusive or text-intensive, the thumbnail data can be derived by text-intensive means: For example, programs exist in the prior art which can be used to pull out what are

US 6,496,206 B1

5

determined to be "key words" of the text document, or the text data can be examined for the presence of proper names or text in a headline-type typeface. More sophisticated programs can even be used to synthesize an abstract of a quantity of text.

The thumbnail data for each icon in the displayed folder is stored in a special "cache folder," and the individual files of thumbnail data are associated with the files in the displayed folder, as at step 102. There will generally be a commonality of file names between the displayed folder and the cache folder; this association of names is particular convenient using the "Shell Name Space Extension" utility. There thus exists, soon after a particular folder is opened and displayed on the GUI, a cache folder which includes files which are essentially immediately available for display.

The thumbnails associated with different files are displayed, according to a preferred embodiment of the present invention, whenever the cursor or other pointing device merely touches or comes within a certain range of the icon associated with a particular file, such as at steps 104. It is preferable to display the thumbnail when the cursor is merely touching the icon, as opposed to requiring, for example, one or two mouse-clicks or equivalent to be provided by the user. Once again, the overall purpose of the thumbnail display is simply to give a "quick glance" at the thumbnail associated with the file. Thus, the size of the thumbnail, in the case of a merely graphically-scaled-down version of a first page of a file, should be just large enough that a casual user could get an overall glimpse of the first page of the file.

In order to maintain an economy of memory using the present invention, the thumbnail data associated with an open folder is preferably erased when there is no longer an immediate need that a particular thumbnail will have to be displayed. According to one embodiment of the present invention, it is possible to cause the cache folder to be emptied or erased when the folder associated with the cache folder at a given time is closed, as at step 106. Alternately, cache folders can remain in existence until the system is shut down, or a timer could be used to erase the cache folder at the end of a work day, or after an elapsed period of time.

Although the overall purpose of the system of the present invention is to provide quick glances at files, in the case of navigating through a quantity of large documents such as multi-page text documents, it may be desirable to provide to a user thumbnails of not only the first page of a text document but also subsequent pages. Thus, according to a particular embodiment of the present invention, a means is provided for enabling the user to take quick glances at further thumbnails of further pages within a multi-page document. Of course, obtaining thumbnail data for subsequent pages of a multi-page document is somewhat at odds with the intention of the invention to keep a relatively small amount of data in the cache folder, so that the cache folder can be quickly generated and accessed. To resolve this dilemma, according to a preferred embodiment of the present invention, the first page of any multi-page document is the only page that is automatically used to derive thumbnail data from any multi-page file in the folder. However, further according to this preferred embodiment, in the case of a thumbnail derived from a multi-page file there is provided with the thumbnail a "flip-button" associated with the thumbnail, which makes available to the user a capability of scrolling through further pages within the multi-page file, as shown at the steps indicated as 108 and 110. With reference to steps 108, if a file is identified as a multipage file at the opening of the folder, the resulting thumbnail data can

6

be flagged. With reference to steps 110, if the flip-button in the thumbnail for the multipage file is indicated, a predetermined number of subsequent pages in the file can be accessed, and suitable thumbnail data, such as a text summary for each of the subsequent pages, can be stored in the cache for viewing in the thumbnail when the flip-button is indicated.

An example of a text-intensive thumbnail, showing extracted proper names and headlines, and with an associated flip-button in the top right corner, is shown at FIG. 4. Activation of this flip-button causes the derivation of thumbnail data (either a graphic reduction of subsequent pages, or the extraction of key words or summaries) of the subsequent pages only as needed. Thus, if there exists multi-page files in the folder, as standard practice only the first page of each file will be used to derive thumbnail data, and thumbnail data from further pages will be derived only if specifically requested by the user, by use of the flip-button. Depending on a particular implementation, activation of the flip-button associated with the thumbnail for a multi-page file can cause derivation of the thumbnail data for the entire rest of the multi-page document, or only a certain predetermined number of further pages, such as up to the next ten pages in the multi-page document. Of course, various graphical functional equivalents to the illustrated flip-button, such as a sliding bar or pull-down menu, will be apparent.

The present invention can be distinguished from certain prior art utilities in key ways. It is well known, such as in the consumer versions of the Macintosh® and Windows™ operating systems to provide "help balloons" which are associated with toolbar buttons, and which display short explanations of the functions of different buttons when a cursor is caused to touch a particular toolbar button. The use of these "help balloons" in this context differs from the present invention in that the text of those help balloons is largely immutable and is always associated with a particular toolbar button. In contrast, with the present invention, the thumbnail data, whether a graphical reduction of a page image, or some other derivation of text data, is created on an ad-hoc basis from a standard method which is applied to what ever files are in a particular folder that is opened; further, according to a preferred embodiment of the invention, once the folder is closed, the thumbnail data is erased, and is re-generated next time the folder is opened. There is a significant functional difference in providing help balloons associated with toolbar buttons, which do not change, and providing what is arguably in effect a help balloon associated with a file, in the context where files constantly come and go through a particular computer.

While the invention has been described with reference to the structure disclosed, it is not confined to the details set forth, but is intended to cover such modifications or changes as may come within the scope of the following claims.

What is claimed is:

1. A method of displaying a thumbnail for at least one electronically-stored file in an electronically-stored folder, comprising:

determining the appearance of at least one icon for at least one file in the folder based on at least one of file name or file type;

displaying the at least one icon for the at least one file in the folder to permit a first identification of data in the at least one file;

deriving thumbnail data from the file in the folder, the thumbnail data being a reduced derivative of at least a portion of the data in the at least one file;

US 6,496,206 B1

7

storing the thumbnail data for the file in a cache;

associating the file in the folder with its stored thumbnail data in the cache; and

when a cursor is positioned to hover over the icon displayed for the file, displaying the thumbnail data associated with the file to permit a second identification of the data in the at least one file in greater detail than the first identification.

2. The method of claim 1, wherein the thumbnail data includes a reduced version of an image in the file.

3. The method of claim 1, wherein the thumbnail data includes a subset of text data in the file.

4. The method of claim 1, wherein the thumbnail data includes data derived from only a first page of the file.

5. The method of claim 1, wherein the thumbnail data is derived upon opening the folder.

6. The method of claim 1, wherein the thumbnail data is displayed by touching the cursor to the icon without a mouse-click.

7. The method of claim 1, wherein the thumbnail data is derived from the file by opening the file as a background operation.

8. The method of claim 1, further comprising:

for files containing multiple pages of data, initially deriving the thumbnail data from only a first page of the file;

when the thumbnail data for the file is displayed, displaying a button for retrieving thumbnail data derived from further pages in the file;

when the button is activated, deriving from the file subsequent thumbnail data, the subsequent thumbnail data being derived from at least one further page in the file; and

storing the subsequent thumbnail data in the cache.

9. The method of claim 1, further comprising:

erasing at least a portion of the cache when the folder is closed.

10. A method of displaying a thumbnail relating to an electronically-stored file in an electronically-stored folder, comprising:

determining the appearance of at least one icon for at least one file in the folder based on at least one of file name or file type;

8

displaying the at least one icon relating to the file to permit a first identification of data in the file; and

when a cursor is positioned to hover over the icon without a mouse-click, displaying the thumbnail relating to the file, the thumbnail being a reduced derivative of at least a portion of the data in the file, to permit a second identification of the data in the file in greater detail than the first identification.

11. The method of claim 10, wherein the thumbnail data includes a reduced version of an image in the file.

12. The method of claim 10, wherein the thumbnail data includes a subset of text data in the file.

13. The method of claim 10, wherein the thumbnail data includes data derived from only a first page of the file.

14. The method of claim 10, wherein the thumbnail data includes a summary of data in the file.

15. The method of claim 10, further comprising:

for files containing multiple pages of data, when the thumbnail data for the file is displayed, displaying a button for retrieving thumbnail data derived from further pages in the file;

when the button is activated, deriving from the file subsequent thumbnail data, the subsequent thumbnail data being derived from at least one further page in the file.

16. The method of claim 10, further comprising:

for each file in the folder, deriving thumbnail data from the file;

storing the thumbnail data for each file in a cache; associating each file with the thumbnail data relating to the file in the cache.

17. The method of claim 16, wherein the deriving step is initiated by opening the folder.

18. The method of claim 10, wherein the thumbnail data from a file is derived by opening the file as a background operation.

19. The method of claim 10, further comprising:

erasing at least a portion of the cache when the folder is closed.

* * * * *

EXHIBIT E



US006594697B1

(12) **United States Patent**
Praitis et al.

(10) **Patent No.: US 6,594,697 B1**
 (45) **Date of Patent: Jul. 15, 2003**

(54) **CLIENT SYSTEM HAVING ERROR PAGE ANALYSIS AND REPLACEMENT CAPABILITIES**

(75) Inventors: **Edward J. Praitis**, Woodinville, WA (US); **Scott Berkun**, Redmond, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/315,311**

(22) Filed: **May 20, 1999**

(51) Int. Cl.⁷ **G06F 15/173**; G06F 15/16

(52) U.S. Cl. **709/225**; 709/229; 709/217; 709/219; 709/227; 714/4

(58) Field of Search 709/217, 225, 709/229, 219, 224, 227; 714/4

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,892,911 A * 4/1999 Ishibashi et al. 709/217
 6,134,680 A * 10/2000 Yeomans 709/219

6,202,087 B1 * 3/2001 Gadish 709/206
 6,269,460 B1 * 7/2001 Snover 714/48
 6,273,622 B1 * 8/2001 Ben-David 709/230
 6,353,855 B1 * 3/2002 Hendren, III 709/220
 6,421,715 B1 * 7/2002 Chatterjee et al. 709/219
 6,421,740 B1 * 7/2002 LeCroy 709/331
 6,438,716 B1 * 8/2002 Snover 714/57

* cited by examiner

Primary Examiner—Saleh Najjar

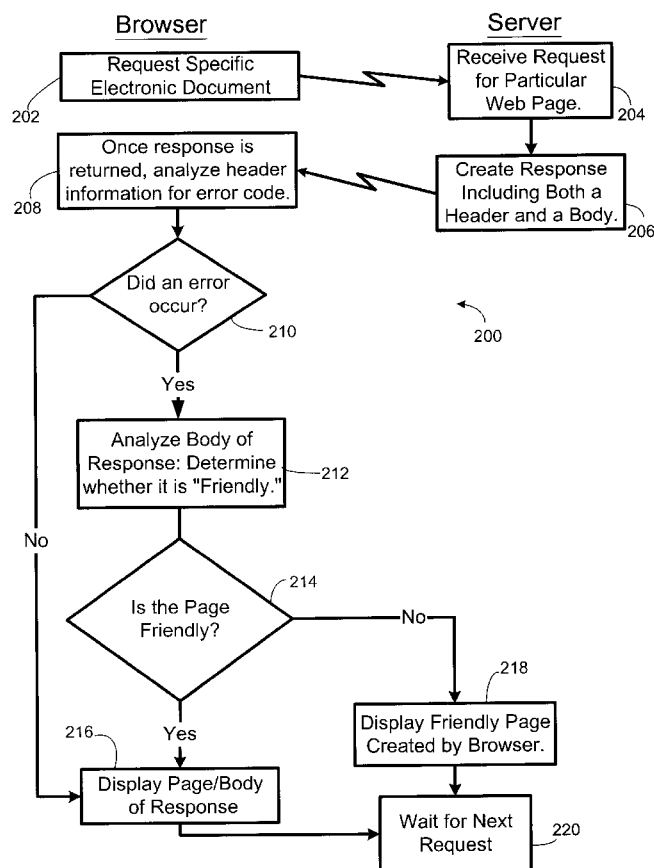
Assistant Examiner—Gregory Garrett Todd

(74) *Attorney, Agent, or Firm*—Merchant & Gould P.C.

(57) **ABSTRACT**

A computer-implemented browser on a client computer that requests electronic documents from a server computer over a computer network and displays friendly error messages or pages when an error is detected. The browser analyzes a response returned to the client computer from the server computer to determine whether an error occurred using information in the header of the response. If an error is detected, the browser analyzes the response to determine whether the response comprises a friendly error page. If not, the browser replaces the page returned in the response with a friendly page in that the browser displays a page separate from the page returned with the response. The replacement page is designed to be more user friendly.

11 Claims, 12 Drawing Sheets



U.S. Patent

Jul. 15, 2003

Sheet 1 of 12

US 6,594,697 B1

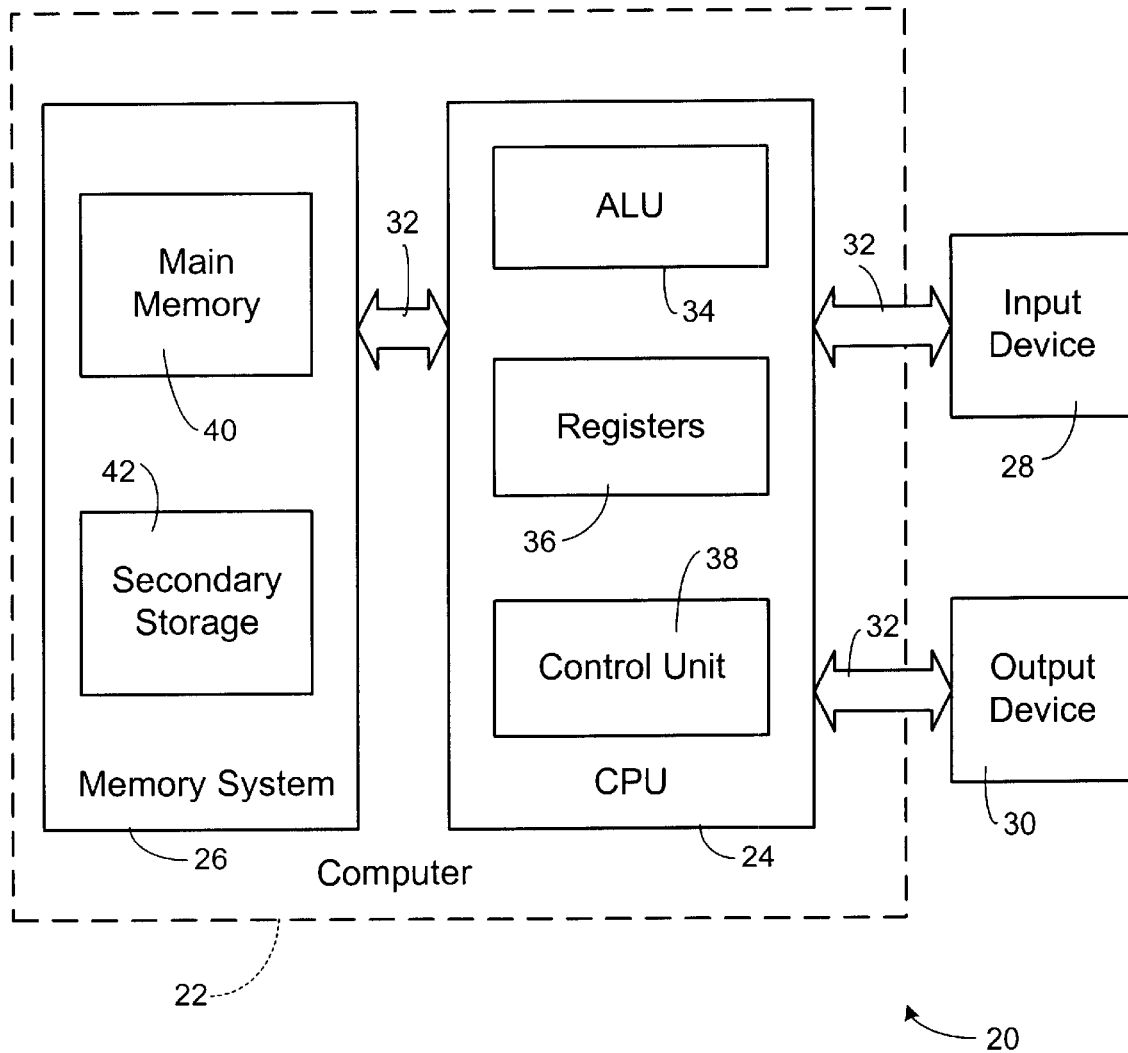
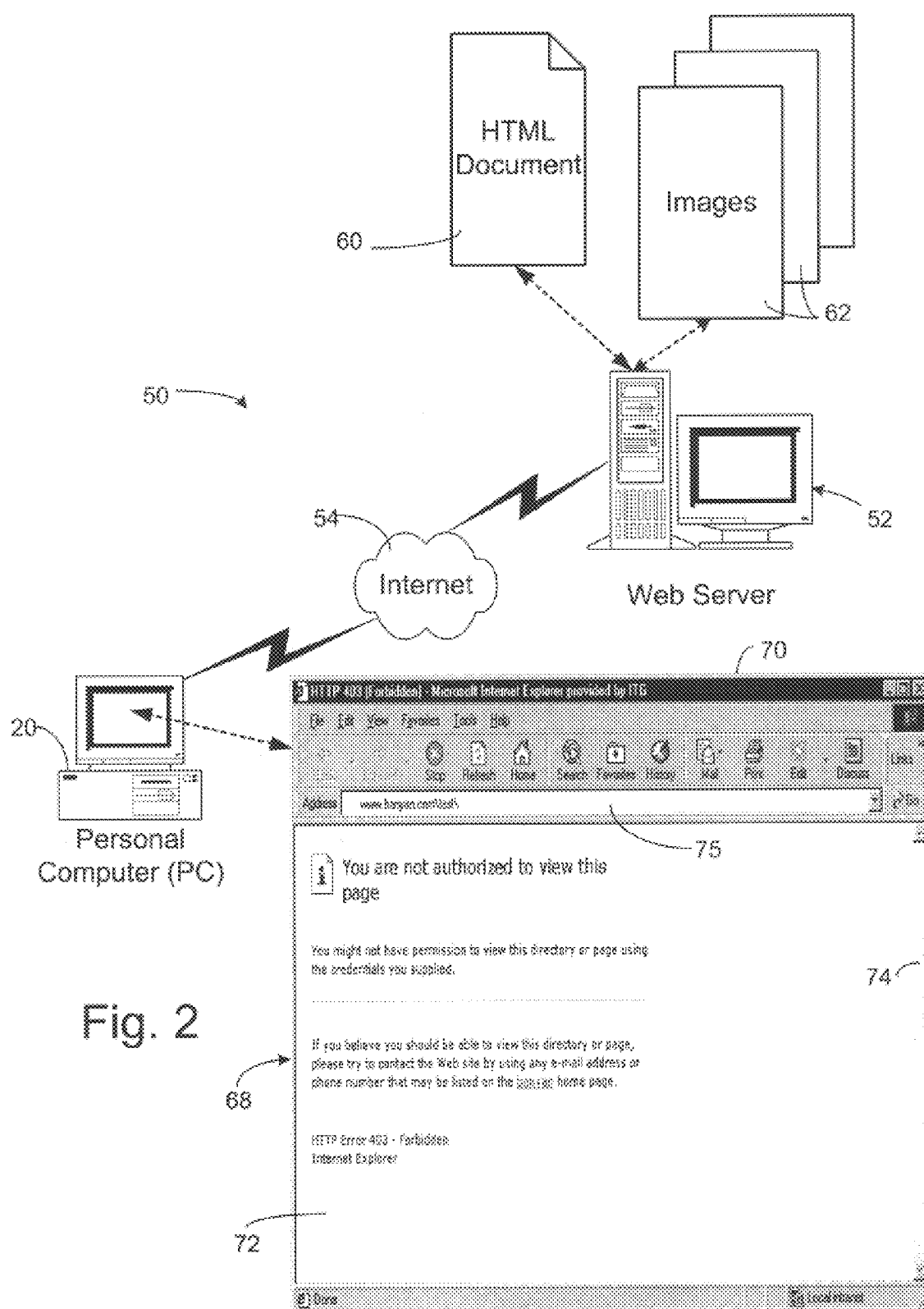


Fig. 1

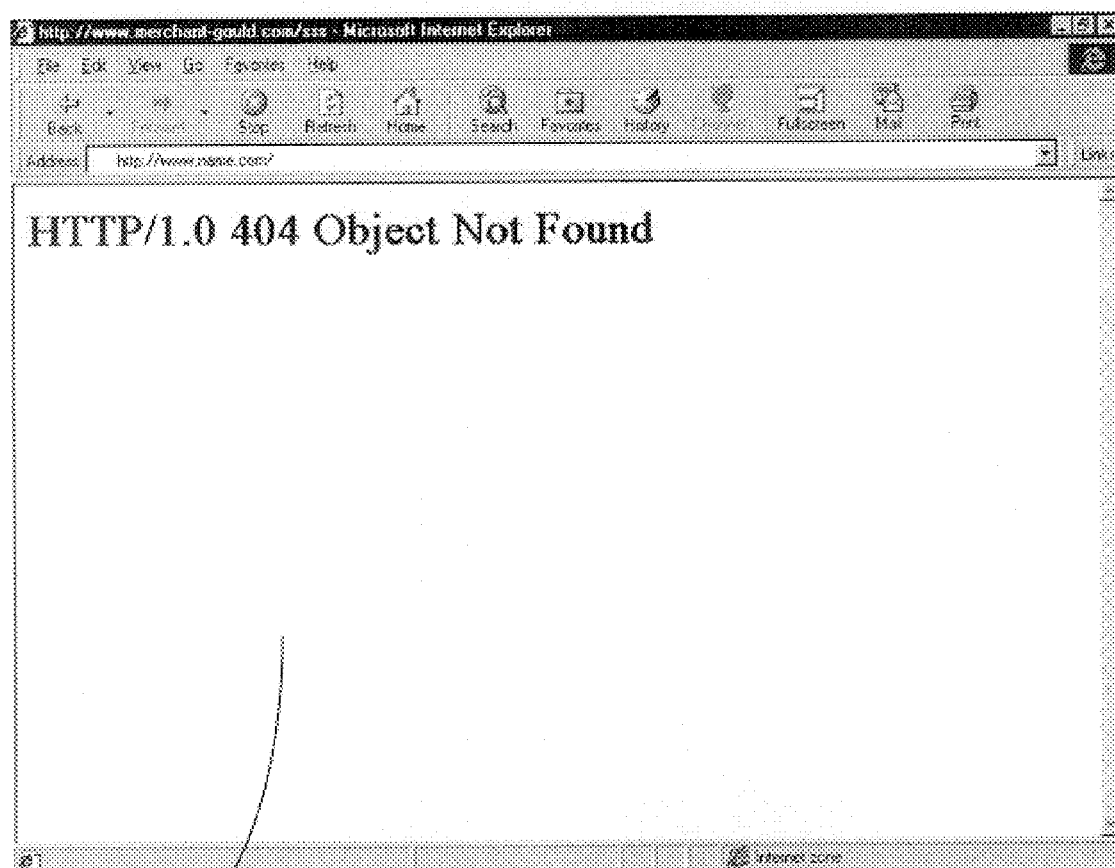


U.S. Patent

Jul. 15, 2003

Sheet 3 of 12

US 6,594,697 B1



76

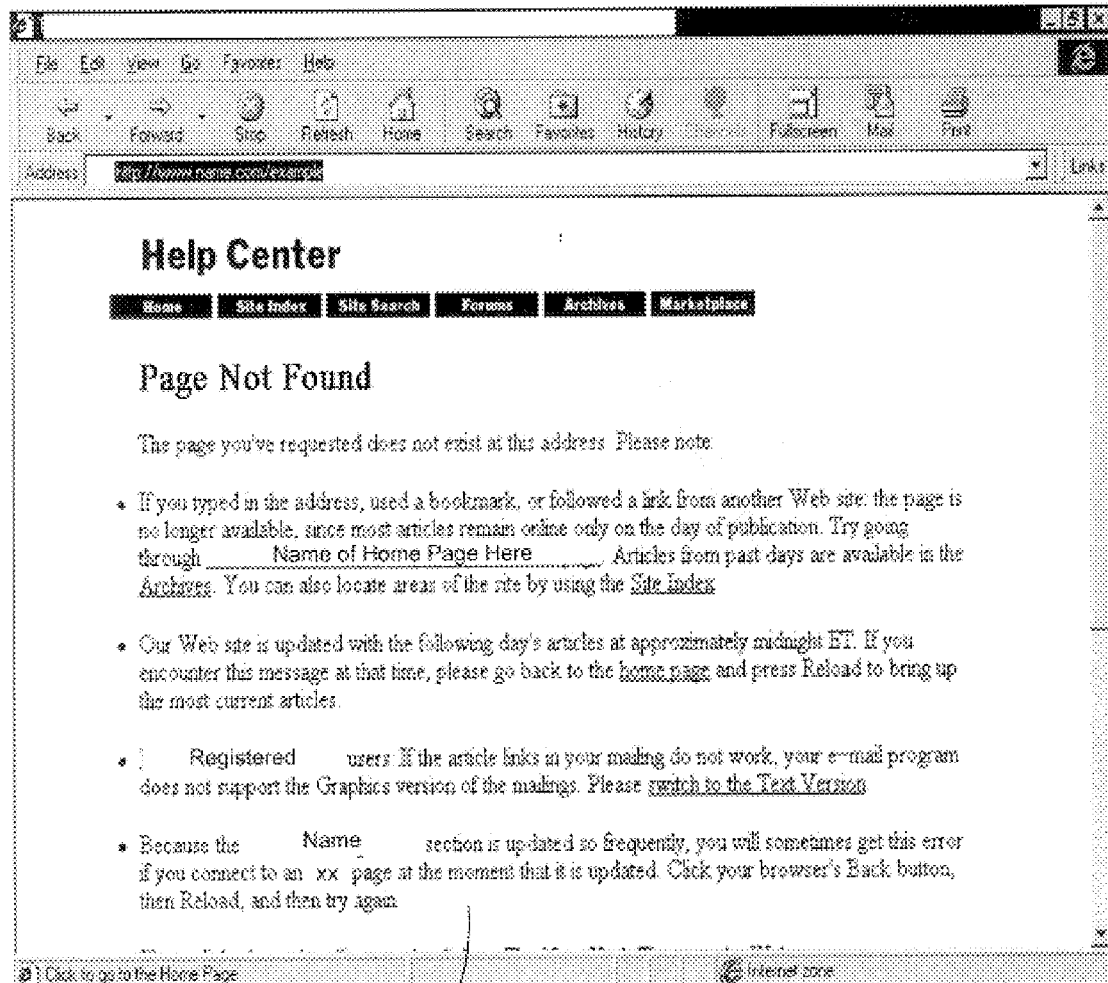
Fig. 3

U.S. Patent

Jul. 15, 2003

Sheet 4 of 12

US 6,594,697 B1



78

Fig. 4

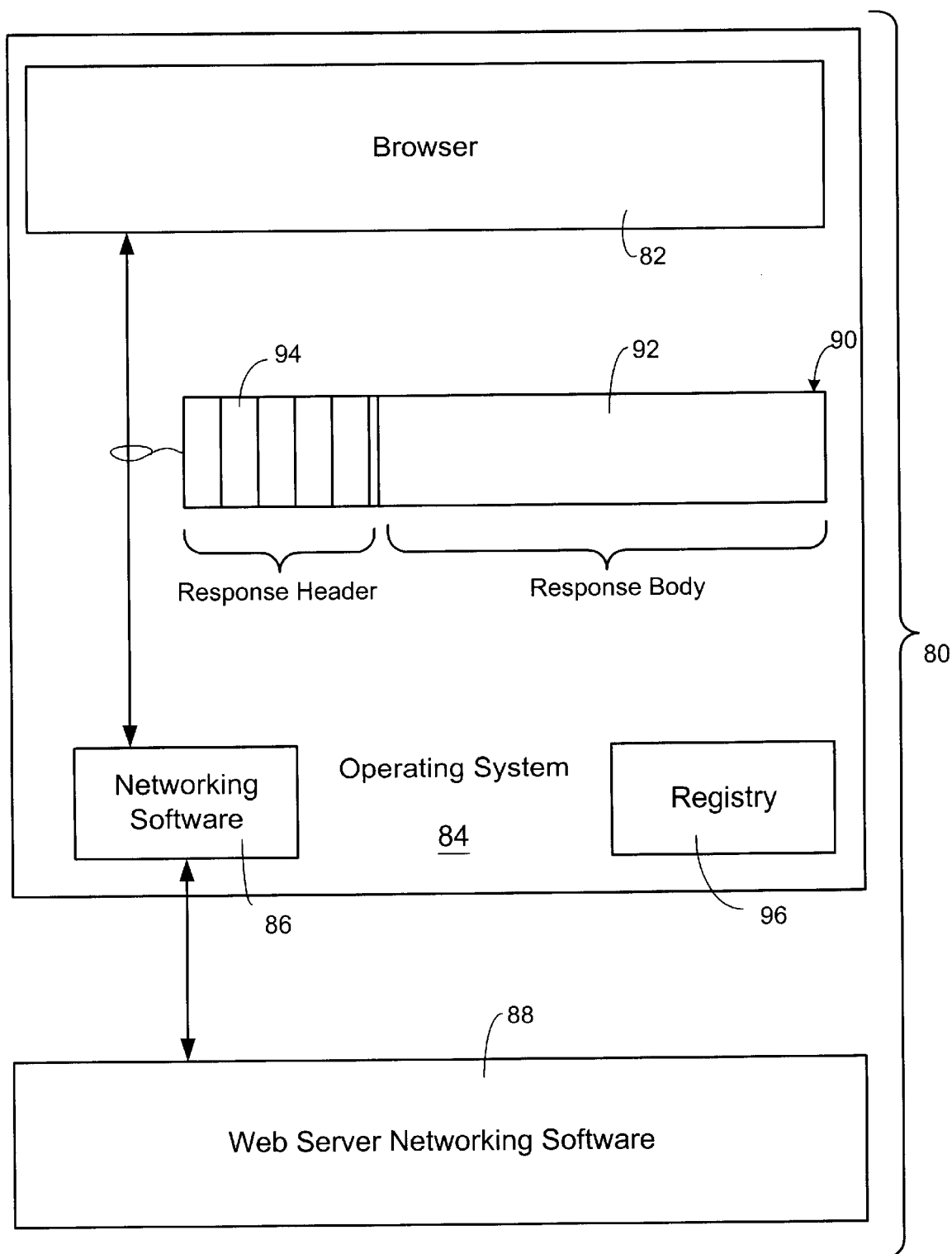


Fig. 5

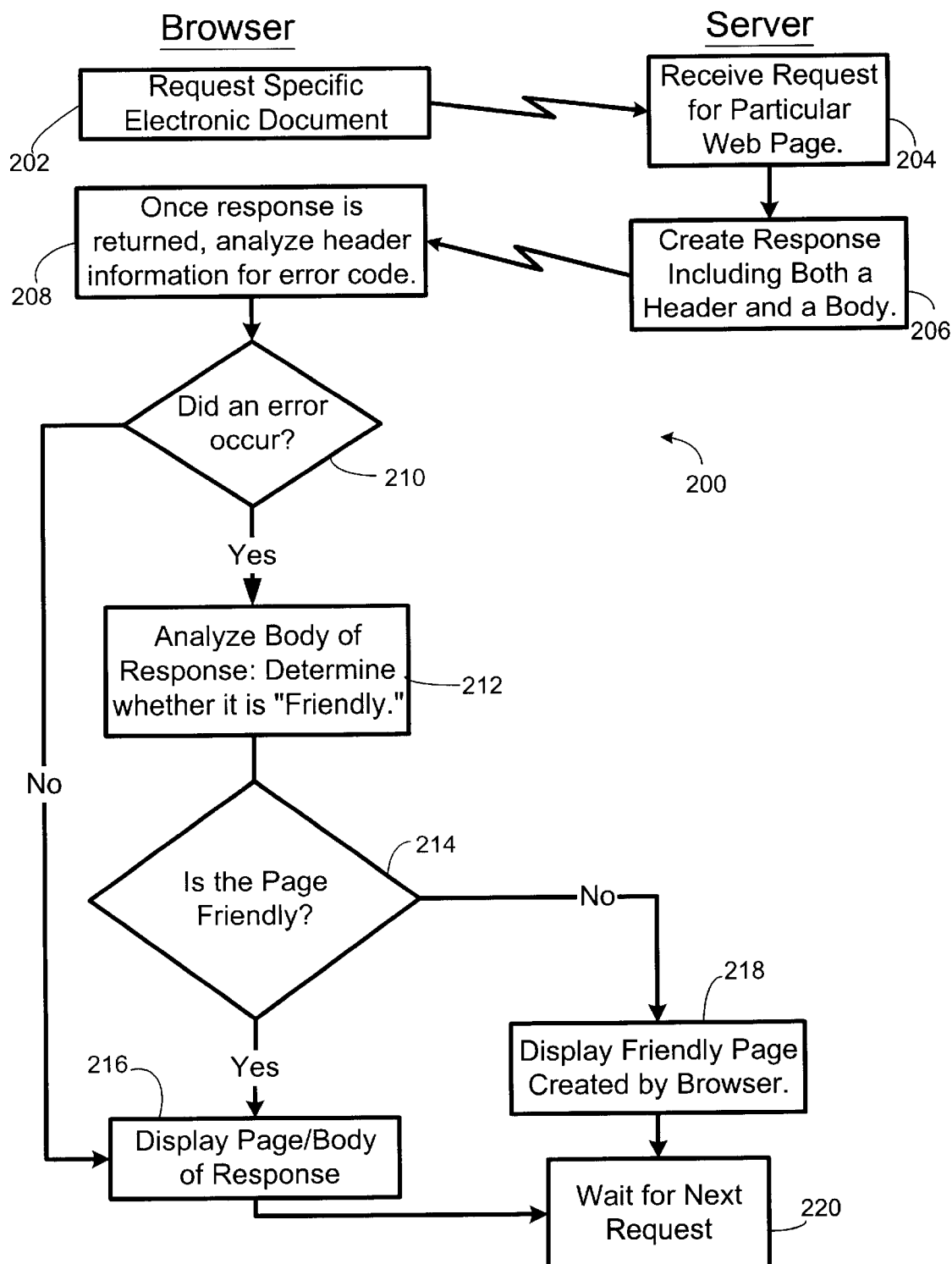
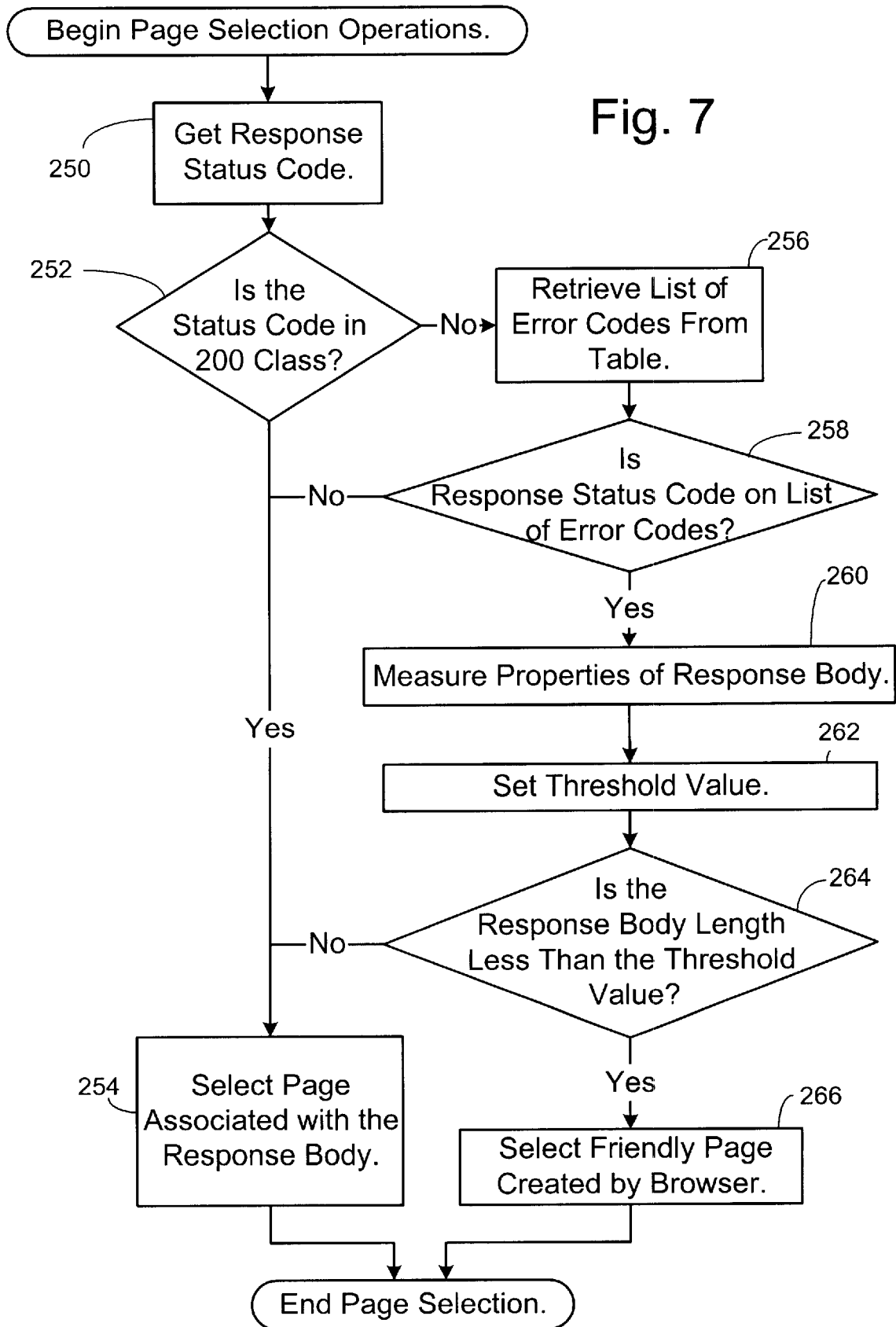
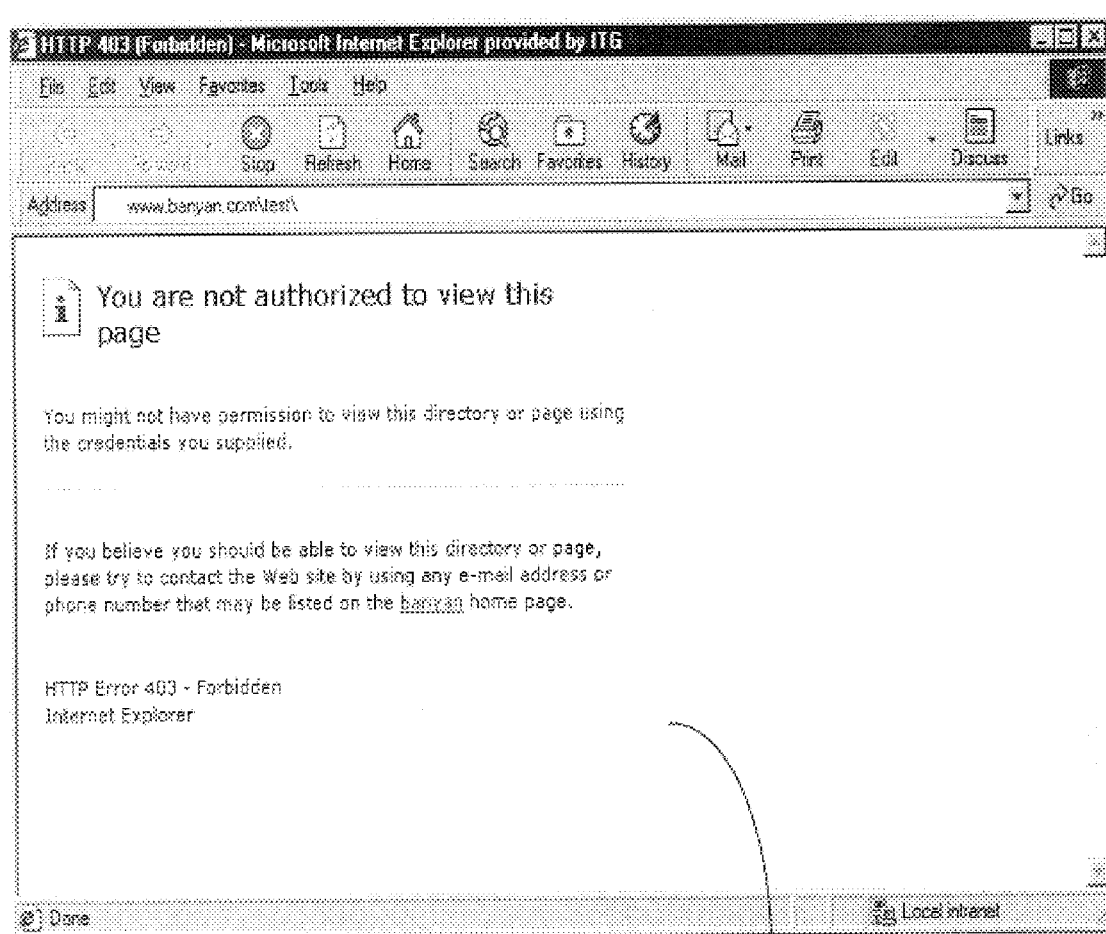


Fig. 6

Fig. 7



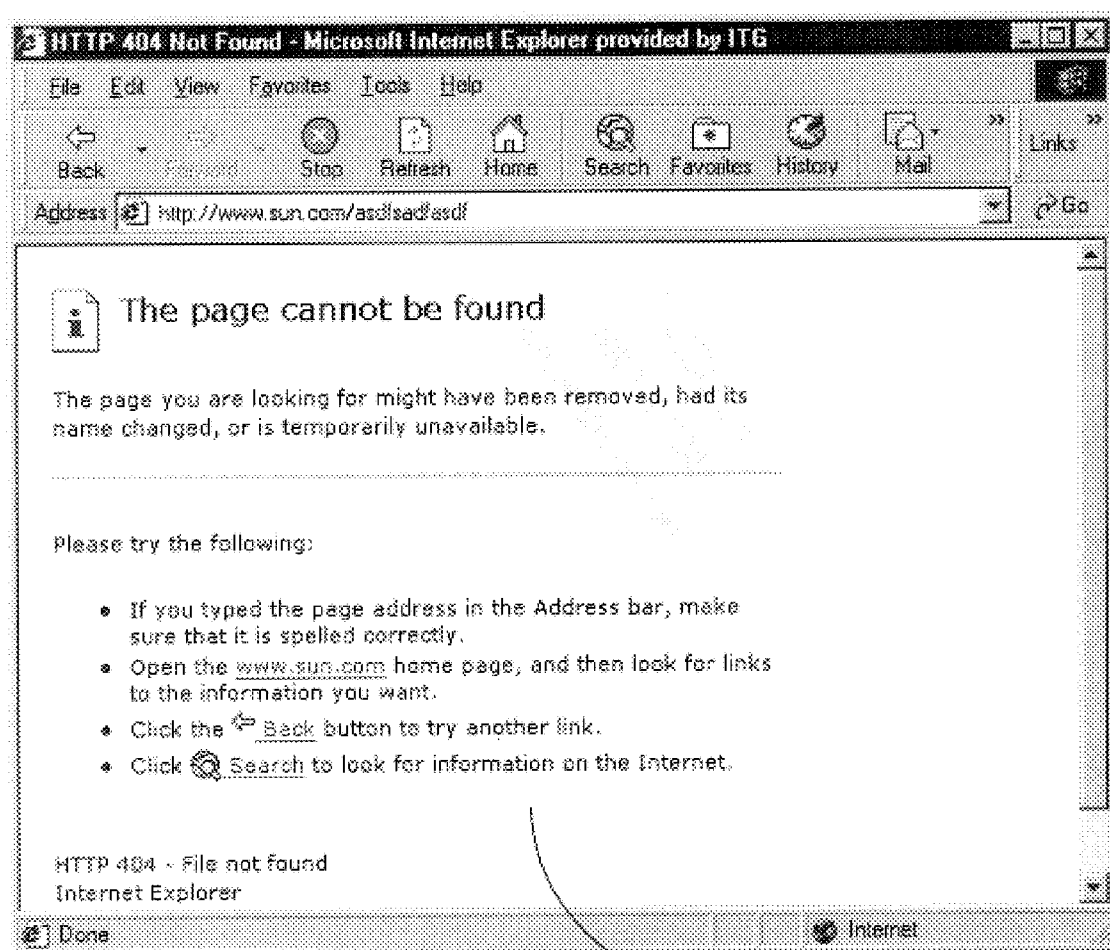
U.S. Patent**Jul. 15, 2003****Sheet 8 of 12****US 6,594,697 B1****Fig. 8**

U.S. Patent

Jul. 15, 2003

Sheet 9 of 12

US 6,594,697 B1



304

Fig. 9

U.S. Patent

Jul. 15, 2003

Sheet 10 of 12

US 6,594,697 B1

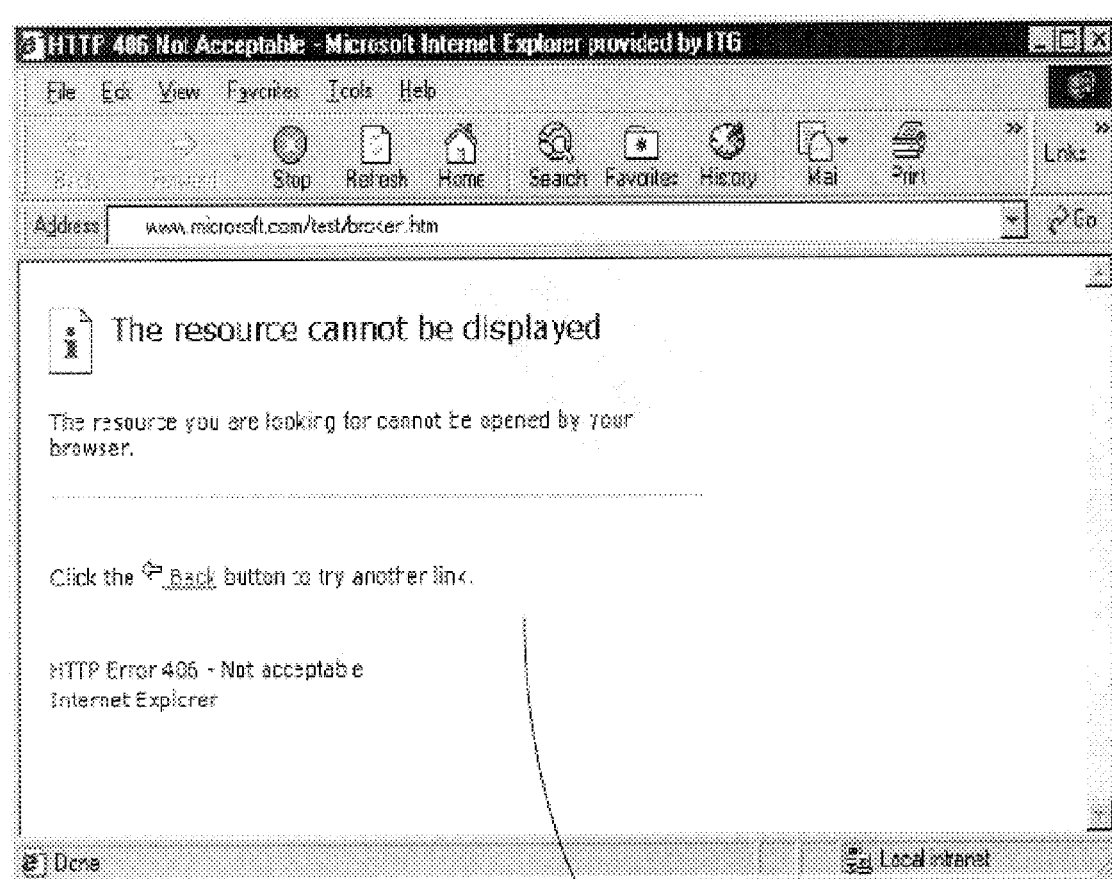


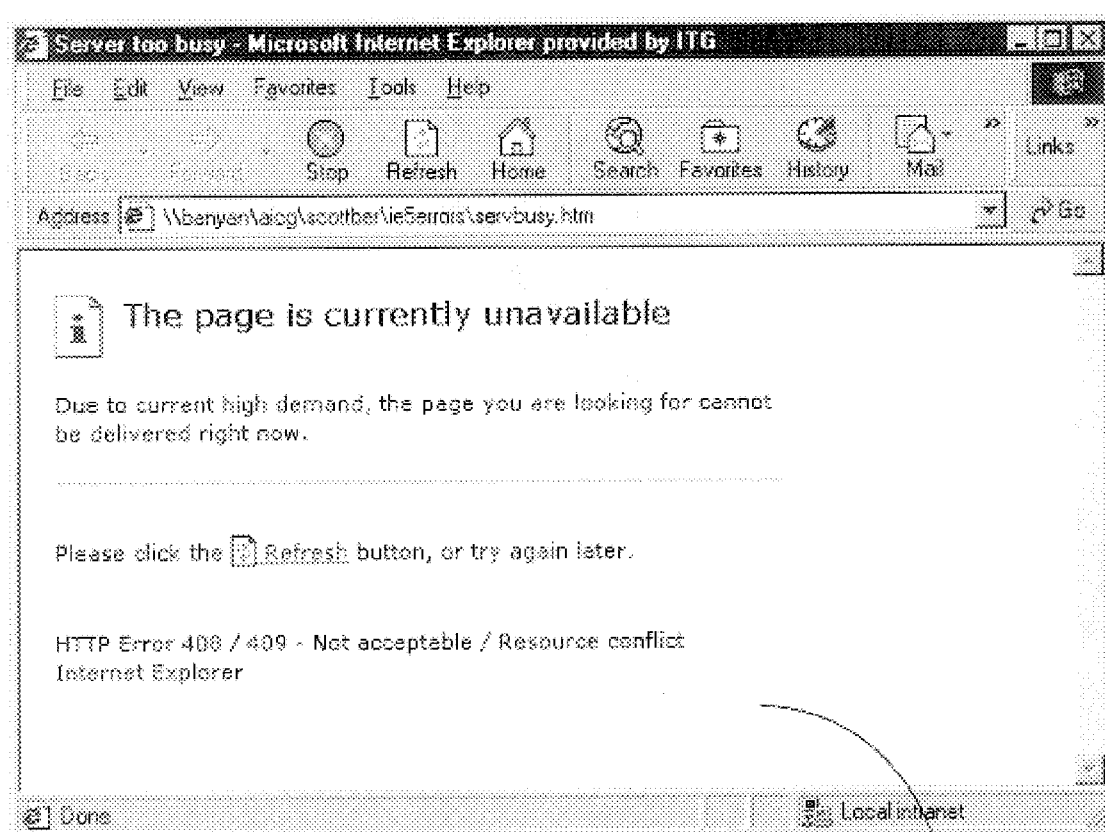
Fig. 10

U.S. Patent

Jul. 15, 2003

Sheet 11 of 12

US 6,594,697 B1



308

Fig. 11

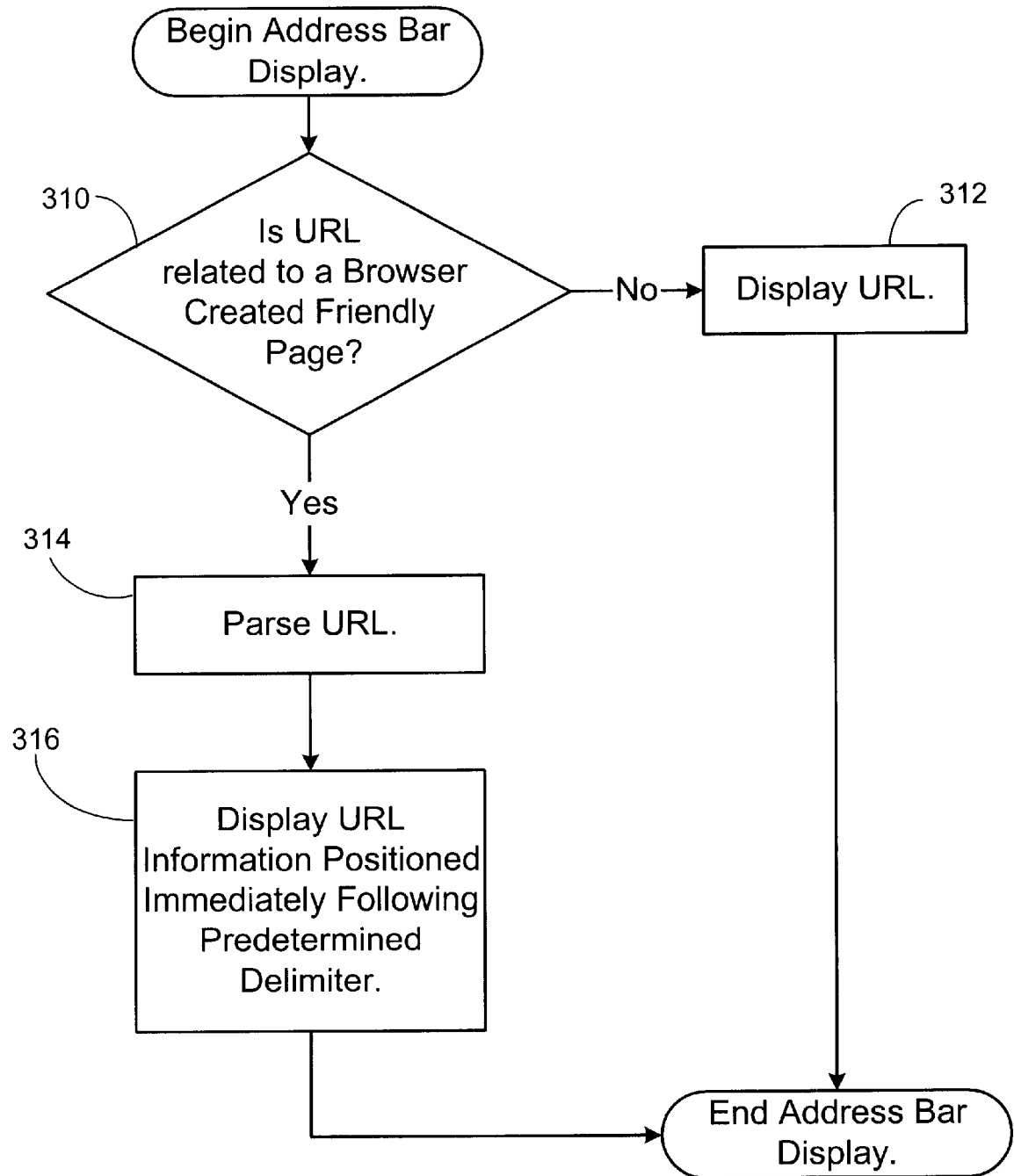


Fig. 12

US 6,594,697 B1

1

CLIENT SYSTEM HAVING ERROR PAGE ANALYSIS AND REPLACEMENT CAPABILITIES

TECHNICAL FIELD

The present invention relates to computer software applications used to facilitate communications between separate computers. More specifically, the present invention relates to software programs, operating on a client computer, that interact with software located on server computers. Further still, the present invention relates to the ability of a client to manipulate a response received by the client from a server.

BACKGROUND OF THE INVENTION

The Internet is a distributed, worldwide computer network comprising computers belonging to various entities such as corporations, institutes of learning, and research organizations. The computers are connected by gateways that handle data transfer and conversion of messages from a sending network to the protocols used by a receiving network. In essence, the Internet is a collection of networks and gateways that use the TCP/IP suite of protocols. TCP/IP is an acronym for transport control protocol/interface program, a software protocol developed by the Department of Defense for communications between computers.

The worldwide web, i.e., web, is a specific Internet network using a specific Internet protocol, i.e., Hyper Text Transfer Protocol (HTTP). In general, protocols are a set of rules in a prearranged data format defining how two computers communicate on the computer network 54. Servers that use the web are known as web servers and typically provide many separate electronic files, displays or documents that are accessible to other web servers or web clients. These electronic files are identified by a uniform resource identifier (URI) or a uniform resource locator (URL).

In connection with wide area networks such as the Internet, operating systems having browsers are available for client computers to facilitate communication between the client computers and the server computer. Browsers in the operating system typically provide many functions, but most importantly they provide a graphical user interface that allows the user to both enter a request for an electronic web document from their personal computer (PC) and to view the response once it is returned.

The process of requesting web documents using the browser is generally referred to as either navigating or browsing the web since it is relatively simple to jump from one web server to the another. Often however, while browsing the web one of many possible errors occurs. Depending on the error, the client system may receive a response informing the client that an error occurred in processing the request. Typically the response will include status code information as to the kind of error but, unfortunately, the information is conveyed to the client system at a relatively low level and few users understand the error. Typically, the information is at such a low level that it is relatively obscure to most users, especially those users not familiar with the specific status codes as defined in the HTTP standards.

As an example, the error displayed message may only provide the information that "error 404" occurred. Receiving information that error 404 occurred does not inform an unsophisticated computer user of anything other than the fact that an error occurred. The user typically has no idea how to proceed or to resolve the issue. Even more experienced computer users may not understand all the highly

2

technical error status codes or all the possible ways to resolve the issue. Unfortunately, receiving such an obscure error message tends to intimidate many computer users.

Some network servers have been proactive in attempting to prevent the display of such highly technical and relatively unfriendly error pages. These servers essentially provide, as part of their response, an electronic document or page display that informs the user of the error that occurred, the possible reasons why the error occurred and possible ways to fix or otherwise avoid the recurrence of the particular error. For example, a web server that receives a request for a particular file or page that is not present on the server may return a web page acknowledging that error 404 occurred, that the file is not found or not located on the server and that the user should double check their request to make sure an error was not made in the request.

Although these pages are much more helpful, servers are not required to provide such pages, and thus most servers do not. Moreover, it appears unreasonable to mandate that all servers comport to rigorous standards requiring these friendly pages. It is with respect to these and other considerations that the present invention has been made.

SUMMARY OF THE INVENTION

The present invention is related to an improved client system having the capability to analyze responses returned from various servers and determine whether an error occurred at the server. Another aspect of the present invention relates to client system having the capability of analyzing the full response associated with an error to thereby determine whether corrective action should be taken. When taken, corrective action preferably relates to replacing the error-response page returned from the server with a client generated page display. Additionally, the client system of the present invention is capable of generating and displaying the replacement page to the user.

The replacement page displayed by the client is designed to be more "user friendly" by providing useful information to the user related to which error occurred, why the error occurred and some possible solutions which may enable the user to either fix the error or avoid the error in the future. In another feature of the invention, the friendly error page also provides hyper-links to areas or sites on the World Wide Web that are considered to be safe, which are those web pages that are both accessible and returnable to the user without an error occurring.

A more complete appreciation of the present invention may be obtained from the accompanying drawings, which are briefly summarized below, from the following detailed description of the invention and from the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a computer system that may be used to implement a method and apparatus embodying the improved browser of the present invention.

FIG. 2 is a block diagram of the computer system shown in FIG. 1 connected to a server computer through a computer network.

FIG. 3 is a reproduction of a sample error page returned by the web server shown in FIG. 2 wherein the page is determined to be unfriendly by the client system of the present invention.

FIG. 4 is a reproduction of a sample error page returned by the web server shown in FIG. 2 wherein the page is determined to be friendly by the client system of the present invention.

US 6,594,697 B1

3

FIG. 5 is block diagram of a software environment comprising the client system of the present invention and a sample response returned to the client system shown in FIGS. 1 and 2 from the web server shown in FIG. 2.

FIG. 6 is a flow chart depicting logical operations executed by the client system of the present invention.

FIG. 7 is a flow chart depicting logical operations executed by a particular embodiment of the client system of the present invention.

FIG. 8 is a reproduction of a sample error page displayed on the screen of the personal computer in FIGS. 1 and 2, the page being created by the client system of the present invention and displayed in substitution for an unfriendly error page.

FIG. 9 is a reproduction of a sample error page displayed on the screen of the personal computer in FIGS. 1 and 2, the page being created by the client system of the present invention and displayed in substitution for an unfriendly error page such as the one shown in FIG. 3.

FIG. 10 is a reproduction of a sample error page displayed on the screen of the personal computer in FIGS. 1 and 2, the page being created by the client system of the present invention and displayed in substitution for an unfriendly error page.

FIG. 11 is a reproduction of a sample error page displayed on the screen of the personal computer in FIGS. 1 and 2, the page being created by the client system of the present invention and displayed in substitution for an unfriendly error page.

FIG. 12 is a flow chart depicting logical operations executed by the client system of the present invention in displaying information in an address bar display area.

DETAILED DESCRIPTION OF THE INVENTION

An embodiment of the present invention displays meaningful, i.e., friendly error messages by analyzing each response returned by the various servers and determining whether to display an electronic document or page returned by the server or to display a predetermined replacement page that is known to be friendly. The header of each response is analyzed for an error indicator such as specific code numbers indicating error conditions. When an error occurs, the length or size of the returned page is compared to a predetermined threshold size. If the returned page is smaller than the threshold size, then the returned page is assumed to be not friendly and is replaced with a friendly page previously stored in memory or created prior to display. In an embodiment, the replacement page is created by the browser. On the other hand, if the returned page is larger than the threshold size, the returned page is assumed to be a friendly error page and is displayed.

Alternatively, the response page is analyzed for other criteria or characteristics that indicate whether the page is friendly. For example, the response page may be analyzed based on the number of hyperlinks or images in the page, the size and type of the page text or the type of software package used by the server computer in generating the response. For any of these measured properties, the threshold value is preferably set to prevent most if not all unfriendly error pages from being displayed while still allowing server generated friendly pages to be displayed.

The predetermined replacement page is preferably generated locally by the operating system having a browser and incorporates information from the requested address URL to

4

provide a hyper-link to a safe page. Additionally, the browser of the operating system preferably displays the URL related to the error URL instead of the friendly page URL in the address bar portion of a browser display so that the user can see which address was requested when the error occurred.

Exemplary Operating Environment

FIG. 1 and the following discussion under this subheading are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. Although, not required, the invention is described in the general context of computer executable instructions such as program modules being executed by a personal computer (PC). Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor based or programmable consumer electronics, network PCs, mini computers, main frame computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network in a distributed computing environment, program modules may be located in both local and remote memory storage devices.

An exemplary system for implementing the invention is shown in FIG. 1. The system comprises a computer system 20 incorporating a computer 22 in the form of a PC that comprises at least one central processing unit (CPU) 24, a memory system 26, an input device 28, and an output device 30. These elements are coupled by at least one system bus 32.

The CPU 24 is of familiar design and includes an Arithmetic Logic Unit (ALU) 34 for performing computations, a collection of registers 36 for temporary storage of data and instructions, and a control unit 38 for controlling operation of the system 20. The CPU 24 may be a microprocessor having any of a variety of architectures including, but not limited to those architectures currently produced by Intel, Cyrix, AMD, IBM, DEC and Motorola.

The system memory 26 comprises a main memory 40, in the form of media such as random access memory (RAM) and read only memory (ROM), and a secondary storage 42 in the form of long term storage mediums such as hard disks, floppy disks, tape, compact disks (CDs), flash memory, etc. and other devices that store data using electrical, magnetic, optical or other recording media. The main memory 40 may also comprise video display memory for displaying images through the output device 30, such as a display device. The memory 26 can comprise a variety of alternative components having a variety of storage capacities such as magnetic cassettes memory cards, video digital disks Bernoulli cartridges, random access memories, read only memories and the like may also be used in the exemplary operating environment. Memory devices within the memory system 26 and their associated computer readable media provide storage of computer readable instructions, data structures, program modules and other data for the computer system 22.

The system bus 32 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures.

US 6,594,697 B1

5

The input and output devices **28** and **30** are also familiar. The input device **28** can comprise a keyboard, a mouse, a microphone, etc. The output devices **30** can comprise a display, a printer, a speaker, etc. Some devices, such as a network interface or a modem can be used as input and/or output devices. The input and output devices **28** and **30** are connected to the computer **22** through system buses **32**.

The computer system **20** further comprises an operating system. The operating system comprises a set of software commands that controls the operation of the computer system and the allocation of resources. Preferably, the operating system employs a graphical user interface where the display output of an application program is presented on the display device **30**. Exemplary operating systems include Microsoft Windows **95**, Microsoft Windows **98** and Microsoft Windows NT operating system. Additionally, the operating system includes a browser module and networking software having capabilities of interacting with other computers over a computer network.

Additionally, the computer system **20** may comprise an application program wherein the application program is the set of software that performs a task desired by the user, using computer resources made available through the operating system. Both are resident in the memory system **26**.

The embodiments of the invention described herein are implemented as logical operations in a distributed processing system or network **50** having a client computer system **20** and at least one network server computer system **52**, as shown in FIG. **2**. The logical operations of the present invention are implemented (1) as a sequence of computer implemented steps running on the computing system **20** and (2) as interconnected machine modules within the computing network **50**. Accordingly, the logical operations executed by the browser portion of the operating system of the present invention as described herein are referred to alternatively as operations, steps or modules.

Client-Server Overview

In the client-server environment **50** of an illustrated embodiment of the invention shown in FIG. **2**, the client computer system **20** runs a browser module (hereinafter browser) as part of the operating system on the computer **20** for retrieving or browsing electronic documents from a remote server computer **52**. The illustrated remote computer network **54** is the Internet, which is described in the background and summary of the invention above. In the illustrated client-server environment **50** the client computer system **20** connects to the computer network **54** over a telephone line with a modem (not shown) or other physical connections alternatively can be used such as a network interface, an ISD1, T1 or the like high speed telephone line, a television cable, a satellite link, an optical fiber network, an Ethernet or local area network technology wire and adapter card, radio or optical transmission devices, etc. The invention can alternatively be embodied in a client-server environment for other public or private computer networks, such as computer network of a commercial on line service or an internal corporate local area network (LAN) or like computer networks.

An electronic document **60** resides at a remote computer **52** also referred to as a web server connected to the computer network **54**. The illustrated electronic document **60** conforms with HTML standards, and may include extensions and enhancements of HTML standards. In conformance with HTML the electronic document **60** can incorporate other additional information content **62**, such as images, audio

6

video executable programs, etc., hereafter simply images **62**, which also reside at the remote computer **58**. The electronic document **60** and images **62** preferably are stored as files in a file system of the remote computer **52**. The electronic document **60** may incorporate the images **62** using HTML tags that specify the location of files containing the images on the Internet **54**. In alternative network protocol embodiments of the invention the electronic document **60** can have other structured document formats.

The browser on the computer **20** retrieves an electronic document **60** from its site, i.e., the web server **52** on the Internet **54**, and displays the document on the computer screen or output device **30** (FIG. **1**). To view the document **60**, the user specifies a URL related to the particular document **60**, such as by entering a URL character string with a keyboard, by selecting a hyperlink specifying the URL in an HTML document currently being displayed in the browser display **68**, or by selecting a URL from a list provided by the browser. In response to the entered URL the browser generates a request command for the URL and transmits the request on the Internet **54** for the document **60** and the respective images **62** related to the document **60** using conventional Internet protocols, such as the Hypertext Transport Protocol (HTTP).

In a preferred embodiment, the browser utilizes a graphical interface, generating the rectangular viewing or display area **68** on the screen of the computer's output device **30** as is conventional in an operating system with a graphical user interface. The browser includes a frame **70** with graphical interface user controls (e.g. menu bar, scroll bars, buttons, etc.) which generally surrounds a document area **72** in the display **68**. The user interface controls for the frame **70** can be activated by the user with the input device **28** to control the browser.

The browser displays the electronic document **60** that the user is currently viewing in the document display area **72**. If the electronic document is too large to completely fit within the document area **72** the browser displays a portion of the document referred to hereafter as the "visible portion" in the document area **72** and presents the scroll bar **74** in the browser frame **70**. The user can manipulate the scroll bar **74** with a mouse or other pointing device or input key commands on the keyboard to change the visible portion of the document that is shown by the browser within the document display area **72**. The display **68** also comprises an address bar **75**. The address bar displays the URL for the document **60** currently being displayed in document area **72**.

Besides HTML documents for browsing, the network server **52** also has HTML documents **60** related to error messages. However, the web server may also incorporate error message documents not in HTML format. The electronic document **60** located at the web server **52** related to error messages may be either a non-friendly error message page or a friendly error message page. A non-friendly is a page that does nothing more than alert the user of an error. A non-friendly error page **76** that has been rendered and replicated is shown in FIG. **3**. Similarly, a sample friendly error page **78**, which provides more information to the user, that has been rendered and replicated is shown in FIG. **4**. These pages can be displayed in the document viewing area **72** of the window **68** shown in FIG. **1**. Importantly, the sample pages **76** and **79** in FIGS. **3** and **4** are pages or documents **60** that are retrieved from a web server **52**. These pages are examples of current practices in communicating errors to the user.

A software operating environment, **80** shown in FIG. **5**, within the client-server environment **50** (FIG. **2**) includes a

US 6,594,697 B1

7

browser module **82** as part of the operating system **84**. In a preferred embodiment, the browser **82** is integrated into the operating system **84** as shown in FIG. 5. Alternatively, the browser **82** could be a separate application which communicates with the operating system **84**.

The operating system **84** also comprises networking software and device driver **86** which implements networking protocols for communicating on the Internet **54** thus indirectly with the network server software **88**. For example, the operating system **84** in the illustrated embodiment may be Microsoft Corporation's Windows **98** operating system, which uses a remote network access subsystem, a TCP/IP network protocol drivers, and a network adapter driver as the networking software **86** for communicating on the Internet. The browser **82** communicates with the networking software **86** using a set of application programming interfaces (APIs) of operating system functions and services to retrieve the electronic document **60** from the web server **52** (FIG. 2).

Following a request for an electronic document **60**, the browser module **82** receives a response **90** shown in FIG. 5 from the networking software module **86** related to the actual response communicated by the web server **52** (FIG. 2). The response **90** comprises both a body **92** and a header **94**. If the request is satisfied, the body **92** comprises the actual electronic document **60** which was requested. Typically, if the request is for a web page comprising text, images and the like, then the body **92** is displayed by the browser **82** on the screen output device **30** (FIG. 1). If the request is not fulfilled due to an error occurring on the server, the response body **92** comprises an electronic text document **60** similar to either the page **76** (FIG. 3) or the page **78** (FIG. 4).

The response header **94** comprises information related to the response, such as the identification of the destination of the response, the origination of the response and the date of the response. Also, each response header **94** returned to the browser comprises a status code number related to the type of response. A successful, satisfactory or ratified request fulfilled by the server generates a header status code number in the 2xx series or class. A response header contains a status code number in either the 4xx or 5xx class when an error occurred in the server while attempting to satisfy the request as defined by the HTTP standards.

In the HTTP standards, client errors are listed under numbers 400–417, referred to as the “4xx class” of errors. The 4xx class of status codes is intended for cases in which an error is returned by the server, but the cause of the error is typically due to an error in the request from the client. The following is a brief list of the status code numbers for the 4xx class of errors identified in the HTTP standards along with their respective error titles and a short explanation of the probable source of the error.

400: Bad Request: The request could not be understood by the server due to improper syntax.

401: Unauthorized: The request requires authorization or additional authorization to be satisfied.

402: Payment Required: Although a valid code, the HTTP standards, as of Nov. 18, 1998 has labeled this code as reserved for future use.

403: Forbidden: The server understood the request, but refuses to satisfy the request.

404: Not Found: The server found nothing matching the requested URL.

405: Method Not Allowed: The method specified in the request is not allowed for the resource identified by the request.

8

406: Not Acceptable: The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the header information sent in the request.

407: Proxy Authentication Required: The client must first authenticate itself with the proxy.

408: Request Time-out: The client did not produce a request within the time that the server was prepared to wait.

409: Conflict: The request could not be completed due to a conflict with the current state of the resource. This code is used in situations where the server expects that the client might be able to resolve the conflict and resubmit the request.

410: Gone: The requested resource is no longer available at the server and no forwarding address is known.

411: Length Required: The server refuses to accept the request without a defined content length of the request message.

412: Precondition Failed: The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server. This response is returned when the client provided some precondition information that prevents the return of the resource in cases when the resource does not satisfy the precondition.

413: Request Entity Too Large: The server is refusing to process a request because the request entity is larger than the server is willing or able to process.

414: Request-URI Too Long: The server is refusing to service the request because the request URL is longer than the server is willing to interpret.

415: Unsupported Media Type: The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

416: Requested Range Not Satisfiable: A server generally returns this response status code when a request included requested range information and none of these range values overlap the current extent of the selected resource.

417: Expectation Failed: An expectation given in the request could not be met by this server, or, if the server is a proxy, the server has unambiguous evidence that the request could not be met by the next-hop server.

Server errors are listed under the numbers 500–505 and are generated by the server when the server is aware that it has erred or is incapable of performing the request. These errors include the following:

500: Internal Server Error: The server encountered an unexpected condition which prevented it from fulfilling the request.

501: Not Implemented: The server does not support the functionality required to fulfill the request, e.g., the server does not recognize the request method.

502: Bad Gateway: The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

503: Service Unavailable: The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

504: Gateway Time-out: The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI.

US 6,594,697 B1

9

505: HTTP Version not Supported: The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

Logical Operations of Preferred Embodiments

In one embodiment of the present invention shown as a flow of logical operations in FIG. 6, the browser module 82 begins with a request operation 202. Operation 202 communicates the request to the web server.

Operation 204 at the server receives the request from the browser. The web server, in module 206, creates a response to the particular request. If the request is satisfied by the server without error, the created response comprises the requested electronic document 60 (FIG. 2) in the body 92 (FIG. 5) of the response. The response header 94 (FIG. 5) also comprises a status code number indicating the request was successful. On the other hand, if the response can not be satisfied due to an error in the request or an error at the server, the server created response at module 206 comprises the appropriate error status code number in the header 94 (FIG. 5) and an error message page as the body 92 of the response 90. Module 206 sends the response to the browser.

Analysis module 208 receives the response from the server. As part of module 208, the information received from the server is passed to the networking software module 86 of the network client computer system. This underlying, lower-level communications application receives the information in packets, recombines the packets into a legible response 90 (FIG. 5) and returns the complete response to the browser module 82. The response 90 contains both the header information 94 and the body 92 which makes up an electronic document or page 60.

Following the receipt of the response from the server, modules 208–214 select the actual page that is to be displayed in the document viewing area 72 (FIG. 2). Initially, the analysis operation 208 analyzes the response header. Decision operation 210 detects, from the analysis operation 208, whether an error occurred in the server. The decision 210 detects errors by comparing information in the response header to predetermine information related to known error conditions. Preferably, the header contains this information in the form of a status code number. If HTTP is used, then the server is required to incorporate the status code number in the response header. Thus, the analysis involves the direct comparison of the status code number returned in the response header to a list of predetermined error numbers resident in the browser. Comparing these numbers allows for a relatively high level determination of whether an error occurred at the server without analyzing the actual body of the response.

If decision operation 210 determines that an error occurred, then the browser operation 212 analyzes the body of the response to determine whether the response body is a friendly response. A friendly response includes a web page having sufficient information to aid the user in handling the error condition. The body analysis operation 212 compares the body of the response to predetermined data characteristics to ascertain whether the returned page is friendly. If it is determined that the response is friendly at decision 214, then display module 216 simply displays the body of the response.

If the body analysis operation 212 determines that the body of the response is unfriendly then the decision operation 214 causes the display operation 218 to display a friendly page instead of the body of the response. The friendly page displayed by operation 218 is created by the

10

browser and is preferably an electronic document file resident on the computer 20. The page is specifically related to the error code status number returned in the header of the response. The end user views a more detailed error message which provides information related to the error that occurred and possible reasons why the error occurred as well as possible ways to avoid the error in the future. Following the display of either the response body or the browser created friendly page, the operating system is free to complete other tasks or wait for another request.

In an embodiment of the present invention, the Internet protocol used is HTTP version 1.1, the logical operations related to the page selection process of this embodiment are shown in FIG. 7 and generally relate to the operations 208–214 described above with respect to FIG. 6. The page selection operations begin following the receipt of a response from the web server 52. Initially, the get operation 250 obtains the response status code. Since the protocol used with this environment is the HTTP version 1.1, the response header includes a number related to the status code. Obtaining the response code involves either making a request to the lower level networking software 86 which then returns the response code to the browser module 82. Alternatively, the browser itself can parse the header information and determine the response status code.

Once the status code is obtained by operation 250, the browser decision operation 252 then tests whether the status code is within the 2xx class of status codes. This test is made by comparing the response code to a list of 2xx class to see if it is one of these codes. Alternatively, the comparison can be done by simply comparing the first digit to the number 2. If the response status code is in the 2xx class, the operation flow branches YES from decision operation 252 to operation 254. This operation is similar to the analysis operation at modules 208 and 210 discussed above in connection with FIG. 6 wherein the specific information is analyzed. The response code 2xx class defined by HTTP version 1.1 outlines the various responses that may be returned by the server when the request was successfully filled.

In such a case no further analysis is necessary related to error messages. Thus, if the browser determines that no error occurred at the server, then the select operation 254 selects the response body as the page to be displayed. Once the response body is selected, then the operation flow is concluded, and the browser continues with other tasks such as displaying selected page and waiting for the next request from the user.

Should the test operation 252 determine that an error occurred at the server during the analysis of the status code information, then operation flow branches NO to retrieve operation 256 which gets a list of error codes from a data table. The data table comprises a predetermined list of error codes, each error code relating to an error which the browser has contemplated as possibly requiring a replacement page.

As discussed above, HTTP outlines approximately 22 different error codes related to many different errors. However, since some of the errors described above rarely occur if ever, the browser does not generate friendly error pages for all HTTP error codes because some of these pages would not be used but would consume memory and impair performance. The following table outlines the more common errors and provides the path to the replacement page:

Error Code	File Name
400	http_400.htm
403	http_403.htm
404	http_404.htm
405	http_gen.htm
406	http_406.htm
408	servbusy.htm
409	servbusy.htm
410	http_410.htm
500	http_500.htm
501	http_501.htm
505	http_501.htm

HTTP Version 1.1 Error Code	Threshold Length Value
400	512
403	256
404	512
405	256
406	512
408	512
409	512
410	256
500	512
501	512
505	512

Once the list is retrieved from the data table, decision operation **258** determines whether the response error code is on the list of error codes. If the response error code is not on the list, then the browser has no friendly error message related to the error. In this situation, select operation **254** at the browser simply selects the response body as the page to be displayed.

If the error code is on the list of error codes then the browser has a friendly error page related to the error and a determination must be made as to whether to select the browser page or the page returned in the response body.

In order to make this determination, the measure properties module **260** analyzes the properties of the response body and produces a property value. Preferably the response is analyzed to determine the length of the response page which produces a length value. Alternatively, the response page may be analyzed for other criteria or characteristics that indicate whether the page is friendly. For example, the response page may be analyzed based on the number of hyperlinks or images in the page, the size and type of the page text or the type of software package used by the server computer in generating the response. Each alternative measurement produces a property value related to the response page.

Obtaining the length of the response body, in terms of bytes, preferably involves executing a command requesting the length value from the networking software **86** (FIG. **5**). Since the networking software **86** compiled the various incoming packets related to the response, the networking software has stored specific information related to the response including the length of the body of the response. Alternatively, some response headers include the length of the response body and thus the information could be gleaned from the header directly. However, since some responses do not include this information, the networking software provides the length.

Once the length of the response body is determined by operation **260**, operation **262** sets a predetermined threshold value. The threshold value is a number related to the expected length of the response body when the response length is measured by module **260**. Since each error condition most likely causes the generation of differently sized error messages, operation **262** sets a threshold value related to the specific error code and independent of the other error codes. A threshold value is thus assigned for each error code condition providing greater flexibility in adjusting threshold values. The following are the preferred values related to the specified error codes:

These values are preferably set according to sizes, in bytes, of existing response bodies containing unfriendly error messages. That is, the threshold size is set to be slightly larger than the largest known unfriendly error message while still being smaller than known friendly pages. Setting the threshold in this manner preferably allows known friendly error messages generated by the network server **52** to be displayed while not displaying unfriendly error messages.

In alternative embodiments that measure properties of the response page other than the length of the response, the threshold values are set in relation to these other property values. The set threshold values preferably prevent unfriendly pages from being selected for display while known friendly server pages may still be selected for display. Also, these threshold values based on other properties may be set according to the error codes.

The threshold values are stored in a registry **96** FIG. **5**, also a part of the operating system **84**. The registry is a data structure used by the operating system **84** to store various items. Since the values are kept in the registry **96**, the values are do not consume memory allocated to the browser. Additionally, placing these values in the registry provides greater flexibility in adjusting the values. For instance, a user may want to adjust the threshold values to allow certain unfriendly error messages to be displayed. In such a situation, the placement of the values in the registry allows the user to configure the threshold values as desired. Of course, the threshold values could be allocated elsewhere while still allowing the user to configure the values.

Upon receiving both the length of the response body and the threshold value for the specific error code, decision operation **264** (FIG. **7**) compares the two values and determines whether the response body length is less than the threshold value. If the body length is not less than the threshold value, then select operation **254** selects the page returned in the response body. In this case, the browser has determined that although an error has occurred and that a browser-created, friendly error page is available, the response body is most likely a friendly error page that should be displayed. The browser assumes that since the returned page is friendly it probably provides more detailed information related to the error and therefore decision operation **264** branches the operation flow NO to select the returned page. Of course, the browser **82** could skip the analysis on the response body, but then friendly pages would not be selected. This may prohibit the best error page from being displayed.

If the response body length is less than the threshold value, the operation flow branches YES to select operation **266** which selects a browser created friendly page. The exact page that is selected is from the table above. Essentially when the match is made between the error code and the list

US 6,594,697 B1

13

from the table, the error message file associated with that error code is stored or marked for later use. Once it is decided that the message will be displayed, the file name is marked so that it can be displayed. Once a page is displayed with respect to a request, then the browser is finished and waits for the next request to be entered by the user.

In accordance with the preferred embodiments of the present invention, the browser has the capability to produce several unique pages **300**, **302**, **304** and **306** as shown in FIGS. **8**, **9**, **10** and **11**. These pages are displayed as substitute pages in place of unfriendly error pages. Other pages (not shown) may be constructed and displayed as is known to those skilled in the art.

In displaying one of the pages **300**, **302**, **304** or **306**, the browser navigates to the page URL. That is, the browser conducts a request similar to the network request that caused the error but instead of requesting a page located on a web server, the browser requests a page located in the memory resident on the computer system **20**. The page is then rendered and displayed in the browser viewing area **72**.

Creating the friendly error pages requires that a script in HTML be constructed in the usual manner. The HTML script incorporates text that is displayed on the screen and provides information related to the error that occurred and potential causes and probable techniques that may be used to avoid the error in the future. These HTML documents also incorporate hyperlinks that can be activated to navigate to safe URL. The browser assumes that the home page related to the web server that was initially accessed is a safe page and thus most error message pages incorporate a hyper link to the home page.

In order to create the hyperlink to the related home page, the HTML document comprises scripted code that parses the requested URL. In parsing the URL, the code searches for a first predetermined delimiter and disregards all information before the delimiter. Next, the code stores each element of the URL up to a second delimiter and disregards all information following the second delimiter. The information between the two delimiters is used to construct a URL which becomes the requested page should the hyperlink become activated. The constructed URL comprises all necessary information such as the proper protocol, delimiters, etc.

The information between the delimiters is also used to create a display code for the hyperlink. The display code displays the information in the form of a hyperlink. Preferably, the information between the delimiters is used as the hyperlink image.

Other hyperlinks are also incorporated that provide similar functions in an attempt to guide the user to a safe page. For example, the page may include a hyperlink which operates to navigate to the URL which was last displayed, similar to activating the "Back" control button in Microsoft Corporation's Internet Explorer Browser program. As another example, the friendly page may include a hyperlink that operates in a similar manner to the Internet Explorer Refresh option.

When a friendly page created by the browser is displayed in the area **72**, the address bar **75**, located near the top of the display area, displays the URL of the page requested from the web server **52**. The browser of the present invention displays the requested URL even though the displayed document is a browser-created error page.

The process of modifying the address bar contents is described in relation to FIG. **12** which is a flow chart of functional operations for the address bar display executed each time a page is displayed in the browser frame.

14

Initially, decision operation **310** determines if the URL related to the page display matches one of the URLs related to the browser created friendly pages. Decision operation **310** thus determines whether a replacement page has been selected for display by the browser. Although a direct comparison is made between the URL to be displayed and a list of browser created URLs, decision operation **310** may be accomplished by setting a flag or otherwise indicating the URL is a replacement page. If no replacement page is used, then the URL does not match any of the browser created URLs and thus operation **312** displays the URL for the page displayed in the browser viewing area **72**.

If the URL matches one of the friendly URLs, then operation **314** parses the URL to determine the initial URL requested. This process may be accomplished by including in the request for the friendly browser page, as a parameter, the initially requested URL. As an example, the request may be of the following form: "res://shdoclc.dll/404.htm#http://www.request.com/file.htm" Wherein the portion of the command that precedes the "#" symbol is the requested friendly page URL. The portion of the request following the "#" symbol is the initially requested URL that returned the error. Passing the initial error URL with the request for the friendly page, allows the parse operation **314** to parse the newly requested URL and select only the URL information related to the initial error URL. Once the error URL is determined, the URL display operation **316** displays the error URL.

Displaying the error URL in the address bar **75** (FIG. **2**) allows the user to immediately view the syntax and spelling of the URL entered which caused the error. Since many errors are related to these types of mistakes, the ability to quickly reference the error URL provides the user with necessary information related to correcting the returned error.

The friendly error pages **300**, **302**, **304** and **306** created and displayed by the browser of the present invention provide the user that encounters them information related to the error that occurred and how it may have happened. The pages also provide the information in a user friendly manner while giving the user several options or techniques by which the user can return to a safe page and avoid the error in the future. As the Internet becomes more popular, the user-friendly error pages provide beneficial guidance to those that would otherwise be overwhelmed by intimidation when faced with the unfriendly error pages.

In an alternative embodiment, the browser may be operated in a text only or non-graphical environment wherein the browser displays browser generated pages of text. The pages of text are considered to be more friendly than returned text error pages.

Presently preferred embodiments of the present invention and many of its improvements have been described with a degree of particularity. It will be understood by those skilled in the art that various other changes in the form and details may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A method of displaying friendly error messages on a client computer following an error occurrence, the client computer communicating with a server computer over a computer network, said method comprising the following steps:

receiving a response from the server wherein the response comprises a response page and an error indicator;
analyzing the error indicator of the response to detect whether an error occurred;

US 6,594,697 B1

15

upon detection of an error, analyzing the response to determine whether the response page is friendly;
 if the response page is friendly, displaying the response page;
 if the response page is unfriendly, displaying a predetermined replacement friendly page;
 wherein the step of analyzing the response to determine whether the response page is friendly further comprises the following steps:
 measuring at least one predetermined property of the response page to generate a property value;
 setting a threshold value;
 comparing the property value to the threshold value; and
 determining whether the response page is friendly based on the comparison of the property value and the threshold value.

2. A method of displaying friendly error messages as defined in claim 1 wherein response page has a response body and the predetermined property is the length of the response body.

3. A method of displaying friendly error messages as defined in claim 1 wherein the predetermined property is a number of hyperlinks in the response page.

4. A method of displaying friendly error messages as defined in claim 1 wherein the predetermined property is a number of images in the response page.

5. A method of displaying friendly error messages as defined in claim 1 wherein the predetermined property is a version of software used by the server in generating the response.

6. An apparatus for providing a friendly error page to a client computing system in a network of server and client computing systems, said apparatus operating in a browser at a client station, said browser sending page requests to the server and the server responding with a response page and response type information, said apparatus comprising:

- a type analysis module analyzing the response type information for an error indication;
- a test module detecting the response type information indicated an error;
- a body analysis module responsive to the error for analyzing the body of the response page to determine if the response page is a friendly error page;
- a select module selecting the response page if the response page is friendly and selecting a generated friendly error page if the response page is not friendly; and

wherein said body analysis module comprises:

- a measure module measuring one or more properties of the response page and creating a measured value; and

16

- a comparison module comparing the measured value to a predetermined threshold value and indicating whether the response page is a friendly error page.

7. The apparatus of claim 6 wherein said body analysis module further comprises:

- a set module setting the predetermined threshold value based on the properties of the response page being measured by said measure module.

8. The apparatus of claim 6 wherein said body analysis module further comprises:

- a set module setting the predetermined threshold value based on a status code.

9. The apparatus of claim 6 wherein said select module comprises:

- detection module detecting whether or not the body analysis module determined the response page was a friendly error page;
- a response page module responsive to the detection module and displaying the response page when the response page is a friendly error page; and
- a created page module responsive to the detection module and displaying a generated friendly error page when the response page is not a friendly error page.

10. A computer data signal embodied in a carrier wave readable by a computing system and encoding a computer program of instructions for executing a computer process for providing a friendly error page to a client computing system in a network of server and client computing systems, said computer process comprising:

- sending page requests to a server;
- receiving a response page and status information from the server;
- analyzing the status information and detecting an error code indicating the response page is an error page;
- if an error code is detected, analyzing the response page to determine if the response page is a friendly error page;
- selecting the response page if the response page is friendly and selecting a created friendly error page if the response page is not friendly;

wherein the act of analyzing the response page comprises:

- measuring at least one property of the response page and creating a measured value; and
- comparing the measured value to a predetermined threshold value and indicating whether the response page is a friendly error page.

11. The computer data signal of claim 10 wherein said computer process further comprises:

- setting the predetermined threshold value based on the response page properties measured by said measuring act.

* * * * *

EXHIBIT F



Visual preview for link traversal on the World Wide Web

Theodorich Kopetzky ^{*,1}, Max Mühlhäuser ¹

Telecooperation Department, Johannes Kepler University Linz, Altenbergerstrasse 69, 4040 Linz, Austria

Abstract

This paper demonstrates a technique for augmenting current World Wide Web browser implementations with features found in classical hypertext applications but unknown to the World Wide Web community until now. An example implementation is shown using Netscape Navigator 4.x using JavaScript, dynamic HTML and Java. The implementation follows an architecture based on a proxy server which acts as a gateway between the Internet and the browsing client. Based on the detailed example, support for further features is discussed. © 1999 Published by Elsevier Science B.V. All rights reserved.

Keywords: Hypertext navigation; User interface; Link preview; World Wide Web

1. Introduction

A novice user of the World Wide Web is usually impressed about the vastness of information available. At a second glance the impression gradually turns into desperation when the user tries to actually find valuable information. The usual solution is to contact a search engine like Altavista [1] or a Web portal like Yahoo [25] which usually work as a starting point for global navigation.

But even when navigating through small Webs (or hyper-documents), users tend to have navigation problems; in essence, these problems boil down to the decision about which links to follow and which to ignore. Considering a state of the art of about 12 years, little improvement was made. It was as early as 1987 that Conklin [9] mentioned the phenomenon in his late introductory article. Zellweger et al. [26] note that it is especially burdensome to follow a link

just to find out that it is not relevant for the reader. They state that the reader leaves the source context just to return to the previous reading position after determining the uselessness of the chosen link. We will briefly review what we consider three levels of support for link navigation.

(1) *Textual hints*. Landow [15] appealed to authors to enrich the context of link anchors in order to help users to decide if a link is worthwhile to follow. Furnas [11] discussed the advantages of links having a scent or residue of contents to come (this may be compared to “highway signage” where the sign shows, e.g., what is next and what is farther away). Zellweger et al. [26] introduce the notion of a *gloss*, which they define as a “brief explanation positioned in the margin or between the lines of a text”. A gloss may contain a description of the target of the link, meta-information and more. Some early hypertext systems like HyperTies [16] or the Guide system [5] already address some issues related to glosses. As to the World Wide Web, the Internet Explorer [17] supports a very basic kind of gloss by optionally

^{*} Corresponding author.

¹ E-mail: {theo,max}@tk.uni-linz.ac.at

displaying a small pop-up window with a textual description of the link to be followed.

(2) *Typed links*. This concise and well-defined remedy to the link navigation problem described has been proposed early on, too. Halasz [13] required typed links, even composite links, in his late “seven issues” article, back in 1988. Manfred Thüning et al. [20] put up eight design principles and ten cognitive design issues to increase global and local coherence of hyperdocuments, one of them being “use typed link *labels*”. However, typed links are (i) not yet well supported, and (ii) not sufficient. Not well supported refers to the fact that HTML 4.0 offers support for link types like “contents”, “chapter”, “next”, etc. [23], but the standard browsers — like Internet Explorer [17] and Netscape Communicator [18] — currently do not support this feature yet. In the context of XML, a special-purpose XML linking language called Xlink [24] is emerging, but its widespread use is still far out. As to the attribute “insufficient”, the limits of typed links are obvious. (i) Unless a commonly accepted ontology is given for a field, the semantics of a given link type are known to the user but not to the reader. The latter must infer these semantics either from the type mnemonics (what might a link of type “in-depth” lead to?) from formal or informal descriptions (which distract the reader even more). (ii) Even if both author and reader have a common understanding of link types, the type information is just one kind of abstraction of the target contents. (iii) Additional authoring effort is required, so that typed links are neither a remedy for the bulk of existing World Wide Web documents nor can they be expected to be soon applied by the myriad of Web designers active in the Internet.

(3) *Further advancements*. Bieber et al. [4] categorize hypermedia environments in four generations in analogy to programming languages (machine languages, assembler languages, high-level programming languages, and finally packages like word processors which let users concentrate on what they want to do). They believe that current authors and readers of the World Wide Web are working in a second-generation hypermedia environment. Further, they discuss a set of hypermedia functionalities which they believe to improve the usability of the Web. One of these functionalities is “global and local

overviews”, to be achieved by overview diagrams and link previews. Obviously, textual hints and typed links as described above are just two facets of link navigation support. Among the more advanced concepts, work on information shape [10] is cited here just as a sample reference. This example illustrates a common problem with hypertext: many concepts have been studied in the past, but they have a long way to go until they are widely available for the World Wide Web community.

In this paper, we describe a technique to implement local overview for Web browsing, called “visual link preview”. The approach falls within category 3 described above (further advancements) and at the same time works with standard browsers without authoring effort; in other words, this approach can be made available to the World Wide Web community immediately.

2. Visual link preview

An author of Web documents has the following options to provide link information to the readers:

- He can describe in the context of the link by text or by image where the link will take the reader. This has been done for example in the image map in Fig. 1.

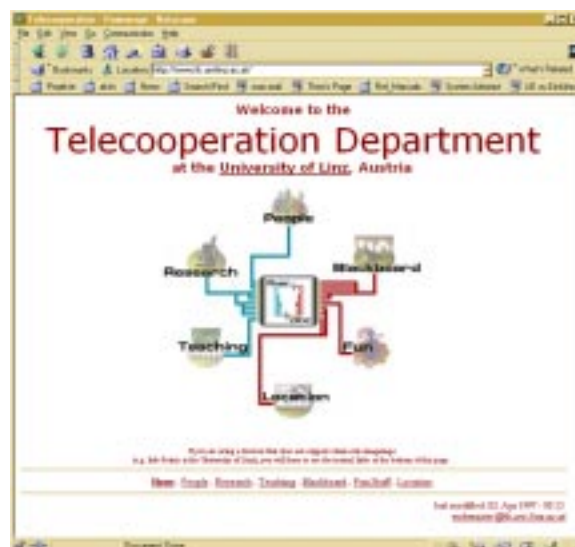


Fig. 1. The homepage of the Telecooperation Department.

- He can use the weak support provided by existing browsers. This support is limited to displaying a small text window.
- He can use dynamic HTML to enrich each link specification and may provide link information in another window or another frame.

All three possibilities demand work to be done by the author. But in the Web environment there is another source of information about the link: the target of the link itself.

Usually the reader does not get to see the link target until he has activated the link, leading to the problems stated in the introduction.

If the reader has already visited a region of the Web, he has built a mental map. Depending on the time passed the reader may need information in addition to the URL to remember that he has already visited this region. This is where the view of the page at the end of the link, provided as thumbnail preview, may help the reader's memory and will aid him in the decision of following the link.

Generating this thumbnail preview by hand can be done for small, static Web sites, but is unfeasible regarding large and fast changing sites on the Web. Thus automatic generation is necessary.

In addition to automatic generation these thumbnails have to meet the following requirements:

- (1) the image should be small to save bandwidth and server space;
- (2) the image should be in color; and
- (3) the quality of the image should be good enough to allow the reader to recall the accompanying image of the target of the link from memory; this is also the reason for point 2.

For the 1st and the 2nd requirement JPEG [14] was chosen as format for the thumbnail images because it provides small image files with a 24 bits color representation.

As for point 3, the quality of the preview images largely depends on the scaling algorithm used. Given the image in Fig. 1, two scaling algorithms are compared.

Below the difference between simple pixel resizing (faster) and bilinear resizing (slower, better quality) can be seen.

To improve recognition the slower method with better results was chosen. In addition, the second

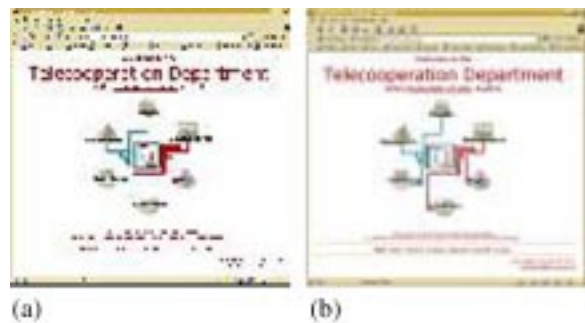


Fig. 2. (a) pixel resizing, 3628 bytes JPEG file. (b) bilinear resampling, 2884 bytes JPEG file.

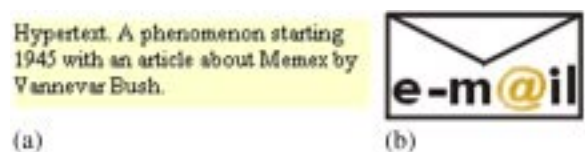


Fig. 3. (a) Possible preview for a link to an anchor in a document. (b) Preview of a link to an E-mail address.

method usually yields smaller image files. To see the original image, take a look at Fig. 1.

2.1. Link types

Although explicit link types are part of HTML 4.0, no standard browser currently supports those.

But links also have implicit types, which can be used for preview purposes. The following list shows which link types are recognized by our system and how they are visualized. As the linking mechanism works with URLs we are using properties of the URLs to categorize a link.

- The URL points to the beginning of a Web page, as in <http://www.tk.uni-linz.ac.at/>. Links of this type are visualized using a thumbnail picture as in Fig. 2b.
- The URL points to an anchor, as in <http://www.encyclopedia.com/h.html#hypertext>². Links of this type may be visualized using a thumbnail picture or, if there is text after the anchor, the text referenced by the link itself may be displayed (see Fig. 3a).
- The URL starts with a protocol different from HTTP, like FTP, TELNET, MAILTO, etc. Al-

² This is a hypothetical link.



Fig. 4. Symbol for a dead link.

though these link types are rather self-explaining a symbol is used to represent the protocol used (see Fig. 3b for an example).

In addition to these properties links can have meta-properties, like *dead* or *forbidden*. If the reader knows in advance that a link is dead, he can safely ignore it. The symbol chosen for a dead link is presented in Fig. 4. These meta-properties are currently derived from the answer of the Web server where the link leads to, and so for example a 404-error is mapped to the dead end symbol.

2.2. Link preview

Fig. 5 shows how the preview looks like using the Netscape Communicator 4.5.

In case the reader requests a preview which is not available yet the reader is informed about that fact at the same location, as seen in Fig. 6.



Fig. 5. The mouse pointer (not shown) moved over the Blackboard symbol and the preview opened.



Fig. 6. No preview available information.

2.3. The presentation of the link preview

A document delivered by the proxy server does not look different to the reader than the original document. The link preview is simply activated by moving the mouse over a link.

The presentation of the link preview is animated. This means that the reader has time to accommodate to the new situation. The preview opens below the link and will remain open for seven seconds.

The preview can be closed by moving the mouse over the preview and then out of it. The preview will close also when the reader moves the mouse over another link and thus activating a new preview image.

To follow the previewed link, the user has the possibility to activate the link itself as usual or to click on the preview image.

3. Architecture

Our aim was to implement link preview without requiring support from the Web author. This approach has the advantage that all existing pages on the Web can be browsed without requiring any change to be made by hand. On the other hand the material has to be changed for the preview to work.

To solve this problem an approach using a proxy server was chosen. An overview of the main components of the proxy server, which has been implemented in Java, can be seen in Fig. 7.

The proxy server has the following tasks:

- analyze the links in a requested HTML document and generate the preview images for all links in the document;



Fig. 7. Basic architecture of the proxy server.

- cache the requested HTML documents and the computed link preview images for future access;
- modify the HTML documents in a way that the requesting browser is able to show the link preview images.

This approach has the following advantages:

- the proxy server has to generate the preview information only once (depending on server space);
- many readers can share one proxy server and thus benefit from already generated preview information;
- readers only have to make one change in their browsing environment: they have to configure the Web client to use a proxy server — everything else is done automatically;
- the proxy server can use other proxies servers and thus benefit from information already fetched from the Web.

3.1. Flow of events

A typical flow of events is described in the following steps (this assumes that in the beginning the required page is cached nowhere).

- (1) The reader enters an URL.
- (2) The browser asks the proxy server for the requested document.
- (3) The proxy server fetches the document from another proxy server or directly from the Web. After the document is fetched completely it is passed to the parser.
- (4) The parser analyses the document regarding its structure and searches for link information. If link information is found, for each link an internal Web browser will be started. These browsers will be used to generate the preview images. The document is modified to enable link preview by inserting the JavaScript code. After parsing is finished the modified document will be cached and passed to the requesting browser.
- (5) The requesting browser displays the document. Link preview images may or may not be ready yet.
- (6) If the reader moves the mouse over a link, the code inserted by the proxy server tries to load the preview image. If it is available at the proxy server, it will be transmitted and displayed, otherwise a notification regarding the missing image will be displayed (see Fig. 6).

Part of these activities happen in parallel to speed up processing.

4. Client-side implementation

To enable link preview at the client side, we had to utilize dynamic HTML. Unfortunately Internet Explorer and Netscape Communicator implement dynamic HTML each in their own way. The Internet Explorer uses cascading style sheets [22], which are not fully supported by Netscape's Communicator and the Communicator uses a technique called layers which is not supported by the Internet Explorer. We decided to implement the first version for the leading browser at that time (which was Netscape).

4.1. Layering

To display the preview information so-called layers are used. Layers can be viewed as small HTML documents which are displayed in the same browser window. Layers are displayed over the base window thus hiding part of it. In addition, layers may overlap other layers.

To enable the browser to show the link preview, the proxy server inserts a layer definition for invisible layers and a short JavaScript program at the beginning of each downloaded HTML-page. An example for a layer definition can be seen in Fig. 8. The position where the insertion actually takes place depends on the structure of the document (if it contains a head tag or a frame set, for example).

Additionally each `<A HREF>` tag is modified to react to events when the mouse pointer moves over the link. A mouseover event-handler is inserted into each link description. This handler will activate a procedure in the inserted JavaScript program if the reader moves his mouse over the link. The handler passes as a parameter the name of the preview image to the procedure. This name was inserted into the link definition just as the event-handler. The procedure will make the invisible layer visible and set some parameter of the layer, e.g. the information which preview image is to be shown.

Fig. 9 shows how a document looks like after modification by the proxy server. First there is the inserted layer and JavaScript code. Then follows the

```

<LAYER NAME= "preBase"
  LEFT="0" TOP="0" width="100" height="100"
  CLIP="0, 0, 100, 100"
  VISIBILITY="hide"
  onMouseOut="deactivate()"
  bgcolor=#FFFFCC>
</LAYER>

```

Fig. 8. A simple layer definition.

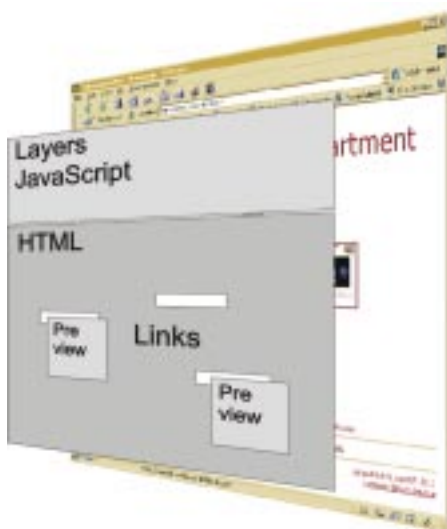


Fig. 9. Client side architecture.

modified HTML code which still may contain links without preview code (e.g. if the parser was not able to parse link information correctly). Preview images are fetched on demand from the proxy server.

5. Limitations

Unfortunately the current implementation has some limitations:

- It only works with the Netscape Communicator. If an Internet Explorer is configured to work with the preview proxy, no preview is available. However, normal usage is not restricted.
- The proxy server needs a lot of bandwidth to build the preview images because for each link encountered and not found in the cache a Web

browser is started. Imagine a reader following a link to a Web page containing a link index with hundreds of links. The proxy server will need some time to build all the preview images.

- The size of each HTML document transferred from the proxy server to the client increases approximately by 2 kb.
- It may interfere with already existing JavaScript and layer code in viewed documents. This can be averted by parsing the document for names used by our system and adapting naming conventions on the fly.

6. Related work

6.1. Classical hypertext

Different types of link preview or displaying the link type exists in several classical hypertext systems, like for example in Sepia [19] which displays the link label next to the link arrow in its link overview diagrams or in HyperTies [16], where a brief description is displayed at the bottom of the screen.

6.2. The Web

There exists an interesting concept which is called displets [21]. Displets provide a principled way to extend HTML. In their proof of concept implementation the authors implemented among others a displet for anchor groups (a way to implement multi-end-point links (see also [4]) and anchor labels. Because displets are implemented as Java classes, they may show any image as well but they lack the server-side component which generates the preview images.

6.3. Other user interfaces

A lot of different approaches exist to display an overview of large information spaces, like Pad++ [3], which uses an interactive panning and zooming navigation metaphor, or distortion-oriented viewing techniques like the generalized fisheye views [12], which employs a distorted representation of out-of-focus material.

Zooming has as disadvantage that the material in focus is moved out of focus if the reader zooms in on a detail. Another disadvantage is that zooming is difficult to integrate in standard Web browsers unless one is willing to implement a plug-in for a browser. Our approach integrates well and requires minimal user intervention to be set up.

Distorting techniques have the same limitations as mentioned with zooming when trying them to integrate in a browser and usually need their own window to work effectively. Our approach reuses the available screen space.

6.4. Link services

An interesting approach to enhance usability of the World Wide Web is the use of a link service [7]. Via link services readers can gain a surplus of information by applying different sets of links to one and the same Web document. Link services store link information external to the documents linked. Current implementations (as shown in [8]) do not store thumbnail information along with the link information. We think that by storing the preview image link services could gain attractiveness.

7. Future work

As already stated the current implementation only works for Netscape 4.x browsers. We are currently working on a version which will include full support for the Microsoft Internet Explorer as well.

In addition we are considering to implement the following features.

- Support for visualizing 1 to n links and selecting them in a screen estate saving way. An example can be seen in Fig. 10 which shows an excerpt

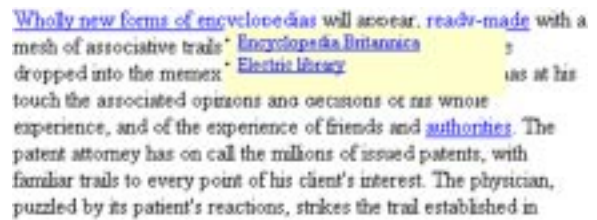


Fig. 10. Excerpt from "As We May think".

from "As We May Think" by Vannevar Bush [6]. A preview window opened below the link "wholly new forms of encyclopedias" revealing two links to choose from. This technique can be easily adapted to display more links.

- Displaying more information about a link. This could be, for example, the ping time to the server where the link is pointing to. Other interesting information could be keywords associated with a link (as long as provided in a format understood by our system).
- More control over the preview. Currently, control is limited to the states "preview on" and "preview off". Users could, for example, define which link to preview in which manner.
- There are some interesting "fluid link" [26] techniques that we are interested in to mimic on the Web.
- As the proxy server approach is basically an intermediary approach as introduced in [2] we are looking into utilizing the power of the Web Browser Intelligence (WBI) architecture.

8. Summary

Visual preview of the target of a link can provide valuable clues for the reader when making the decision if to follow the link or not.

We have shown that a simple preview service can be implemented using standard Web browsing software and described an implementation which requires minimal user activities to be set up and does not need additional client software.

We are continuing to develop our approach adopting to new technologies as they become available and integrating novel navigating concepts.

Acknowledgements

I would like to thank Shai Kariv, Noam Ambar, and Amir Gur for letting me use the source code of their basic proxy server, and Ronald Angerer and Andreas Schwediauer for implementing parts of the system.

References

- [1] AltaVista, <http://www.altavista.com/>
- [2] R. Barrett and P.P. Maglio, Intermediaries: new places for producing and manipulation Web content, in: Proc. 7th Int. World Wide Web Conference, Comput. Networks and ISDN Systems 30 (1998) 509–518.
- [3] B.D. Bederson and J.D. Hollan, Pad++: a zooming graphical interface for exploring alternate interface physics, in: Proc. ACM UIST '94, 1994.
- [4] M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian and H. Oinas-Kukkonen, Fourth generation hypermedia: some missing links for the World Wide Web, International Journal of Human–Computer Studies 47 (1997) 31–65.
- [5] P.J. Brown, Turning ideas into products: the Guide system, in: Proc. ACM Hypertext '87, 1987, pp. 33–40.
- [6] V. Bush, As we may think, Atlantic Monthly 176 (1945) 101–108, <http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm> or <http://www.ps.uni-sb.de/~duchier/pub/vbush/>
- [7] L.A. Carr, D. De Roure, W. Hall and G. Hill, The distributed link service: a tool for publishers, authors and readers, in: Proc. 4th Int. World Wide Web Conference, Boston, MA, USA, O'Reilly and Associates, December 1995.
- [8] L.A. Carr, D. De Roure, W. Hall, G. Hill, Implementing an open link service for the World Wide Web, World Wide Web Journal 1 (1998) 2.
- [9] J. Conklin, Hypertext: an introduction and survey, IEEE Computer (September 1987) 17–41.
- [10] A. Dillon, M. Vaughan, It's the journey and the destination — shape and the emergent property of genre in evaluating digital documents, New Review of Hypermedia Multimedia 3 (1997) 91–106.
- [11] G.W. Furnas, Effective view navigation, in: Proc. ACM CHI '97, Human Factors in Computing Systems, pp. 367–374.
- [12] G.W. Furnas, Generalized fisheye views, in: Proc. ACM CHI '86, 1986, pp. 367–374.
- [13] F.G. Halasz, Reflections on notecards: seven issues for the next generation of hypermedia systems, Commun. ACM 31(7) (1988) 836–852.
- [14] JPEG, Joint Photographic Experts Group, <http://www.jpeg.org>
- [15] G.P. Landow, Relationally encoded links and the rhetoric of hypertext, in: Proc. ACM Hypertext '87, 1987.
- [16] G. Marchionini, B. Shneiderman, Finding facts vs. browsing knowledge in hypertext systems, IEEE Computer (January 1988) 70–80.
- [17] Microsoft, Internet Explorer 4.01, <http://www.microsoft.com/windows/ie/ie40/>
- [18] Netscape, Netscape Communicator 4.x, <http://developer.netscape.com/>
- [19] N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt, M. Thüring, SEPIA: a cooperative hypermedia authoring environment, in: Proc. 4th ACM Conf. on Hypertext, Systems I, 1992, pp. 11–22.
- [20] M. Thüring, J.M. Haake, J. Hannemann, Hypermedia and cognition: designing for comprehension, Communications of the ACM 38(8) (1995).
- [21] F. Vitali, C.-M. Chiu, M. Bieber, Extending HTML in a principled way with displets, in: Proc. 6th WWW Int. Conference, Computer Networks and ISDN Systems 29(8–13) (1997) 1115–1128.
- [22] World Wide Web Consortium, Cascading style sheets, <http://www.w3.org/Style/>
- [23] World Wide Web Consortium, HTML 4.0 Links: document relationships: the LINK element, <http://www.w3.org/TR/REC-html40/struct/links.html#h-12.3>
- [24] World Wide Web Consortium, XML XLink Requirements 1.0, <http://www.w3.org/TR/NOTE-xlink-req/>, February 1999.
- [25] Yahoo, <http://www.yahoo.com/>
- [26] P.T. Zellweger, B.-W. Chang and J.D. Mackinlay, Fluid links for informed and incremental link transitions, in: Proc. ACM Hypertext '98, 1998, pp. 50–57.



Theodorich Kopetzky is a Ph.D. candidate at the Telecooperation department at the Johannes Kepler University of Linz. He received his master of Computer Sciences in 1993 and worked afterwards in the field of embedded systems. He began to work as a researcher at Telecooperation department in 1996. His research interests are hypermedia authoring, navigation, and web-based training.



Max Mühlhäuser is a full professor of computer science and head of the Telecooperation Research Group at Linz University, Austria. His research domain is software engineering for distributed multimedia systems, mobile and ubiquitous computing, and human–human and human–computer cooperation. He received his diploma and PhD. in computer science from Karlsruhe University, Germany, in 1981 and 1986, respectively. He is a member of ACM, the German Computer Society, and IEEE.

EXHIBIT G

OTHER PUBLICATIONS

- Amir, Yair, et al. "Seamlessly Selecting the Best Copy from Internet-Wide Replicated Web Servers." Department of Computer Science, Johns Hopkins University, Jun. 1998, 14 pgs.
- Bestavros, Azer. "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems." In *Proceedings of ICDE '96: The 1996 International Conference on Data Engineering*, Mar. 1996, 4 pgs.
- Carter, J. Lawrence, et al. "Universal Classes of Hash Function." *Journal of Computer and System Sciences*, vol. 18, No. 2, Apr. 1979, pp. 143–154.
- Chankhunthod, Anawat, et al. "A Hierarchical Internet Object Cache." In *Usenix Proceedings*, Jan. 1996, pgs. 153–163.
- Cormen, Thomas H., et al. *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1994, pgs. 219–243, 991–993.
- Deering, Stephen, et al. "Multicast Routing in Datagram Internetworks and Extended LANs." *ACM Transactions on Computer Systems*, vol. 8, No. 2, May 1990, pgs. 85–110.
- Devine, Robert. "Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm." In *Proceedings of 4th International Conference on Foundations of Data Organizations and Algorithms*, 1993, pgs. 101–114.
- Grigni, Michelangelo, et al. "Tight Bounds on Minimum Broadcasts Networks." *SIAM Journal of Discrete Mathematics*, vol. 4, No. 2, May 1991, pgs. 207–222.
- Gwertzman, James, et al. "The Case for Geographical Push-Caching." *Technical Report HU TR 34-94*(excerpt), Harvard University, DAS, Cambridge, MA 02138, 1994, 2 pgs.
- Gwertzman, James, et al. "World-Wide Web Cache Consistency." In *Proceedings of the 1996 USENIX Technical Conference*, Jan. 1996, 8 pgs.
- Feeley, Michael, et al. "Implementing Global Memory Management in a Workstation Cluster." In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, 1995, pgs. 201–212.
- Floyd, Sally, et al. "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing." In *Proceeding of ACM SIGCOMM'95*, pgs. 342–356.
- Fredman, Michael, et al. "Storing a Sparse Table with $O(1)$ Worst Case Access Time." *Journal of the Association for Computing Machinery*, vol. 31., No. 3, Jul. 1984, pgs. 538–544.
- Karger, David, et al. "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web." In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, May 1997, pgs. 654–663.
- Litwin, Withold, et al. "LH—A Scaleable, Distributed Data Structure." *ACM Transactions on Database Systems*, vol. 21, No. 4, Dec. 1996, pgs. 480–525.
- Malpani, Radhika, et al. "Making World Wide Web Caching Servers Cooperate." In *Proceedings of World Wide Web Conference*, 1996, 6 pgs.
- Naor, Moni, et al. "The Load, Capacity and Availability of Quorum Systems." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pgs. 214–225.
- Nisan, Noam. "Pseudorandom Generators for Space-Bounded Computation." In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, May 1990, pgs. 204–212.
- Palmer, Mark, et al. "Fido: A Cache that Learns to Fetch." In *Proceedings of the 17th International Conference on Very Large Data Bases*, Sep. 1991, pgs. 255–264.
- Panigrahy, Rina. *Relieving Hot Spots on the World Wide Web*. Massachusetts Institute of Technology, Jun. 1997, pgs. 1–66.
- Peleg, David, et al. "The Availability of Quorum Systems." *Information and Computation* 123, 1995, 210–223.
- Plaxton, C. Greg, et al. "Fast Fault-Tolerant Concurrent Access to Shared Objects." In *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, 1996, pgs. 570–579.
- Rabin, Michael. "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance." *Journal of the ACM*, vol. 36, No. 2, Apr. 1989, pgs. 335–348.
- Ravi, R., "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pgs. 202–213.
- Schmidt, Jeanette, et al. "Chernoff-Hoeffding Bounds for Applications with Limited Independence." In *Proceedings of the 4th ACS-SIAM Symposium on Discrete Algorithms*, 1993, pgs. 331–340.
- Tarjan, Robert Endre, et al. "Storing a Sparse Table." *Communications of the ACM*, vol. 22, No. 11, Nov. 1979, pgs. 606–611.
- Vitter, Jeffrey Scott, et al. "Optimal Prefetching via Data Compression." In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, Nov. 1991, pgs. 121–130.
- Wegman, Mark, et al. "New Hash Functions and Their Use in Authentication and Set Equality." *Journal of Computer and System Sciences* vol. 22, Jun. 1981, pgs. 265–279.
- Yao, Andrew Chi-Chih. "Should Tables be Sorted?" *Journal of the Association for Computing Machinery*, vol. 28, No. 3, Jul. 1981, pgs. 615–628.
- Beavan, Colin "Web Life They're Watching You." *Esquire*, Aug. 1997, pgs. 104–105.
- Beavan, Colin "Web Life They're Watching You." *Esquire*, Aug. 1997, pp. 104–105.

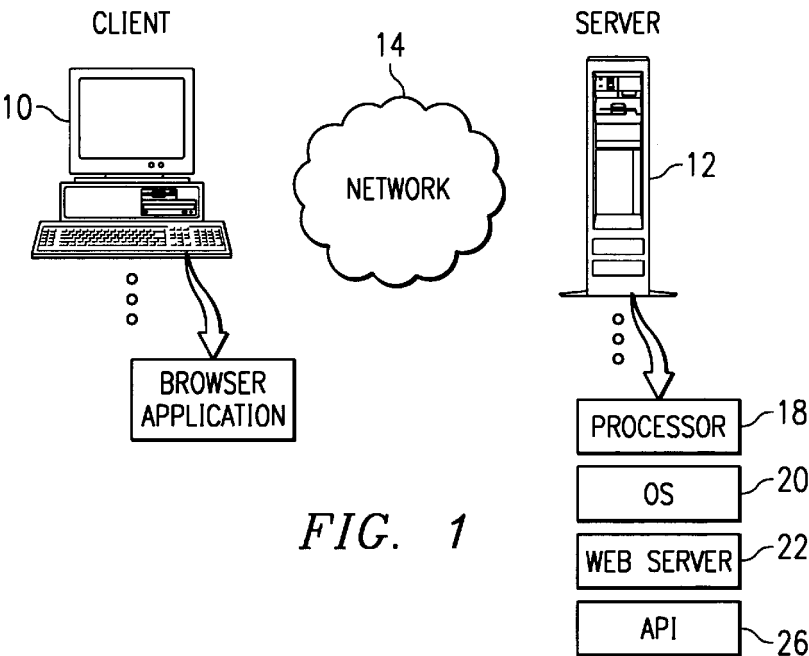


FIG. 1

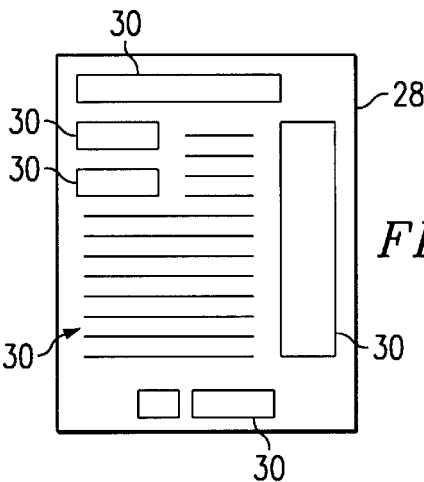


FIG. 2

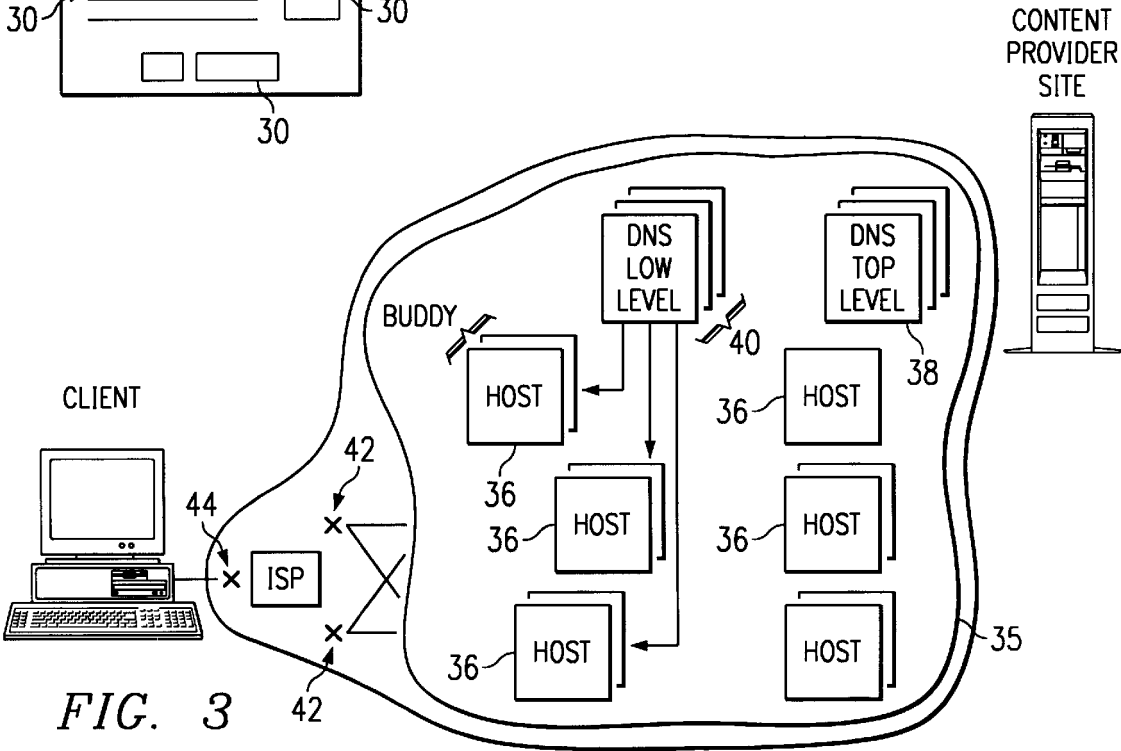


FIG. 3

FIG. 4

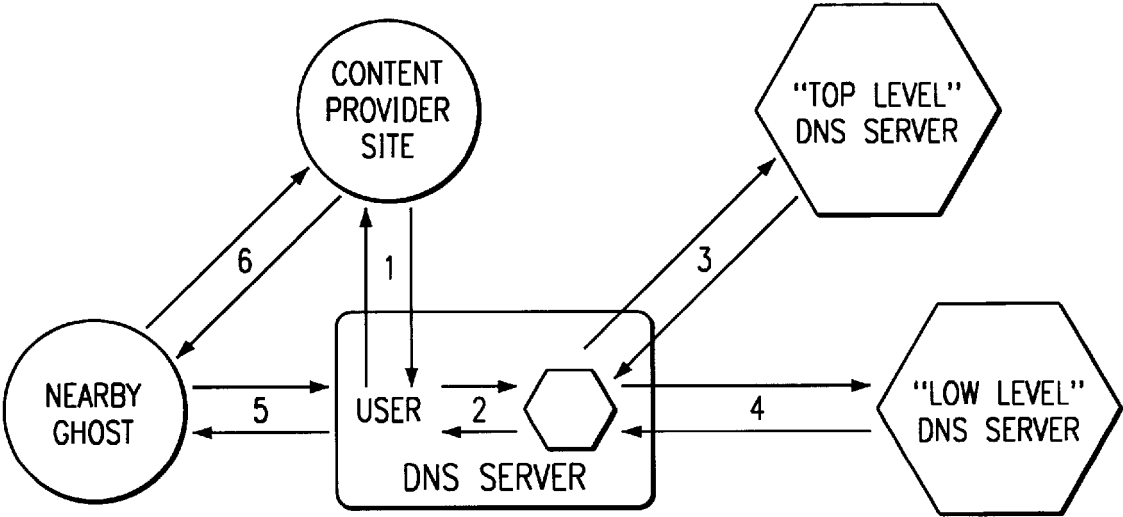
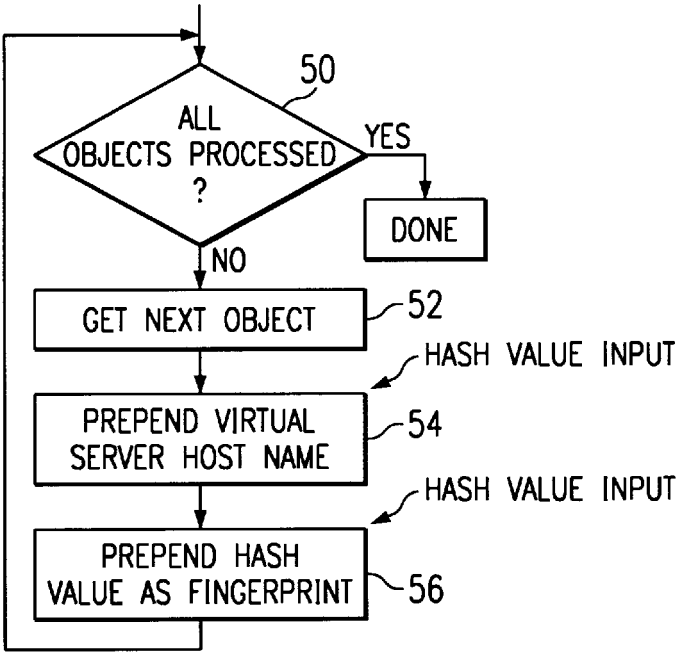


FIG. 5

GLOBAL HOSTING SYSTEM

This application is based on Provisional Application No. 60/092,710, filed Jul. 14, 1998. This application includes subject matter protected by copyright.

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates generally to information retrieval in a computer network. More particularly, the invention relates to a novel method of hosting and distributing content on the Internet that addresses the problems of Internet Service Providers (ISPs) and Internet Content Providers.

2. Description of the Related Art

The World Wide Web is the Internet's multimedia information retrieval system. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and, in return, receives a document or other object formatted according to HTML. A collection of documents supported on a Web server is sometimes referred to as a Web site.

It is well known in the prior art for a Web site to mirror its content at another server. Indeed, at present, the only method for a Content Provider to place its content closer to its readers is to build copies of its Web site on machines that are located at Web hosting farms in different locations domestically and internationally. These copies of Web sites are known as mirror sites. Unfortunately, mirror sites place unnecessary economic and operational burdens on Content Providers, and they do not offer economies of scale. Economically, the overall cost to a Content Provider with one primary site and one mirror site is more than twice the cost of a single primary site. This additional cost is the result of two factors: (1) the Content Provider must contract with a separate hosting facility for each mirror site, and (2) the Content Provider must incur additional overhead expenses associated with keeping the mirror sites synchronized.

In an effort to address problems associated with mirroring, companies such as Cisco, Resonate, Bright Tiger, F5 Labs and Alteon, are developing software and hardware that will help keep mirror sites synchronized and load balanced. Although these mechanisms are helpful to the Content Provider, they fail to address the underlying problem of scalability. Even if a Content Provider is willing to incur the costs associated with mirroring, the technology itself will not scale beyond a few (i.e., less than 10) Web sites.

In addition to these economic and scalability issues, mirroring also entails operational difficulties. A Content Provider that uses a mirror site must not only lease and manage physical space in distant locations, but it must also buy and maintain the software or hardware that synchronizes and load balances the sites. Current solutions require Content Providers to supply personnel, technology and other items necessary to maintain multiple Web sites. In summary,

mirroring requires Content Providers to waste economic and other resources on functions that are not relevant to their core business of creating content.

Moreover, Content Providers also desire to retain control of their content. Today, some ISPs are installing caching hardware that interrupts the link between the Content Provider and the end-user. The effect of such caching can produce devastating results to the Content Provider, including (1) preventing the Content Provider from obtaining accurate hit counts on its Web pages (thereby decreasing revenue from advertisers), (2) preventing the Content Provider from tailoring content and advertising to specific audiences (which severely limits the effectiveness of the Content Provider's Web page), and (3) providing outdated information to its customers (which can lead to a frustrated and angry end user).

There remains a significant need in the art to provide a decentralized hosting solution that enables users to obtain Internet content on a more efficient basis (i.e., without burdening network resources unnecessarily) and that likewise enables the Content Provider to maintain control over its content.

The present invention solves these and other problems associated with the prior art.

BRIEF SUMMARY OF THE INVENTION

It is a general object of the present invention to provide a computer network comprising a large number of widely deployed Internet servers that form an organic, massively fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users.

Another more general object of the present invention is to provide a fundamentally new and better method to distribute Web-based content. The inventive architecture provides a method for intelligently routing and replicating content over a large network of distributed servers, preferably with no centralized control.

Another object of the present invention is to provide a network architecture that moves content close to the user. The inventive architecture allows Web sites to develop large audiences without worrying about building a massive infrastructure to handle the associated traffic.

Still another object of the present invention is to provide a fault-tolerant network for distributing Web content. The network architecture is used to speed-up the delivery of richer Web pages, and it allows Content Providers with large audiences to serve them reliably and economically, preferably from servers located close to end users.

A further feature of the present invention is the ability to distribute and manage content over a large network without disrupting the Content Provider's direct relationship with the end user.

Yet another feature of the present invention is to provide a distributed scalable infrastructure for the Internet that shifts the burden of Web content distribution from the Content Provider to a network of preferably hundreds of hosting servers deployed, for example, on a global basis.

In general, the present invention is a network architecture that supports hosting on a truly global scale. The inventive framework allows a Content Provider to replicate its most popular content at an unlimited number of points throughout the world. As an additional feature, the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location. Moreover, content is automatically sent to the location where it is requested, without any effort or overhead on the part of a Content Provider.

3

It is thus a more general object of this invention to provide a global hosting framework to enable Content Providers to retain control of their content.

The hosting framework of the present invention comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers nearest the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

The determination of which hosting server to use to serve a given embedded object is effected by other resources in the hosting framework. In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality. When a client machine issues an HTTP request to the Web site for a given Web page, the base HTML document is served from the Web site as previously noted. Embedded objects for the page preferably are served from particular hosting servers identified by the top- and low-level DNS servers. To locate the appropriate hosting servers to use, the top-level DNS server determines the user's location in the network to identify a given low-level DNS server to respond to the request for the embedded object. The top-level DNS server then redirects the request to the identified low-level DNS server that, in turn, resolves the request into an IP address for the given hosting server that serves the object back to the client.

More generally, it is possible (and, in some cases, desirable) to have a hierarchy of DNS servers that consisting of several levels. The lower one moves in the hierarchy, the closer one gets to the best region.

A further aspect of the invention is a means by which content can be distributed and replicated through a collection of servers so that the use of memory is optimized subject to the constraints that there are a sufficient number of copies of any object to satisfy the demand, the copies of objects are spread so that no server becomes overloaded, copies tend to be located on the same servers as time moves forward, and copies are located in regions close to the clients that are requesting them. Thus, servers operating within the framework do not keep copies of all of the content database. Rather, given servers keep copies of a minimal amount of data so that the entire system provides the required level of service. This aspect of the invention allows the hosting scheme to be far more efficient than schemes that cache everything everywhere, or that cache objects only in pre-specified locations.

The global hosting framework is fault tolerant at each level of operation. In particular, the top level DNS server returns a list of low-level DNS servers that may be used by the client to service the request for the embedded object. Likewise, each hosting server preferably includes a buddy server that is used to assume the hosting responsibilities of its associated hosting server in the event of a failure condition.

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel

4

technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. More generally, the virtual server hostname is inserted into the URL. Preferably, the virtual server hostname includes a value (sometimes referred to as a serial number) generated by applying a given hash function to the URL or by encoding given information about the object into the value. This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames. In addition, a given fingerprint value for the embedded object is generated by applying a given hash function to the embedded object itself. This given value serves as a fingerprint that identifies whether the embedded object has been modified. Preferably, the functions used to generate the values (i.e., for the virtual server hostname and the fingerprint) are applied to a given Web page in an off-line process. Thus, when an HTTP request for the page is received, the base HTML document is served by the Web site and some portion of the page's embedded objects are served from the hosting servers near (although not necessarily the closest) to the client machine that initiated the request.

The foregoing has outlined some of the more pertinent objects and features of the present invention. These objects should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Many other beneficial results can be attained by applying the disclosed invention in a different manner or modifying the invention as will be described. Accordingly, other objects and a fuller understanding of the invention may be had by referring to the following Detailed Description of the Preferred Embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in connection with the accompanying drawings in which:

FIG. 1 is a representative system in which the present invention is implemented;

FIG. 2 is a simplified representation of a markup language document illustrating the base document and a set of embedded objects;

FIG. 3 is a high level diagram of a global hosting system according to the present invention;

FIG. 4 is a simplified flowchart illustrating a method of processing a Web page to modified embedded object URLs that is used in the present invention;

FIG. 5 is a simplified state diagram illustrating how the present invention responds to a HTTP request for a Web page.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A known Internet client-server system is implemented as illustrated in FIG. 1. A client machine 10 is connected to a Web server 12 via a network 14. For illustrative purposes, network 14 is the Internet, an intranet, an extranet or any other known network. Web server 12 is one of a plurality of servers which are accessible by clients, one of which is illustrated by machine 10. A representative client machine includes a browser 16, which is a known software tool used to access the servers of the network. The Web server supports files (collectively referred to as a "Web" site) in the form of hypertext documents and objects. In the Internet

paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL).

A representative Web server 12 is a computer comprising a processor 18, an operating system 20, and a Web server program 22, such as Netscape Enterprise Server. The server 12 also includes a display supporting a graphical user interface (GUI) for management and administration, and an Application Programming Interface (API) that provides extensions to enable application developers to extend and/or customize the core functionality thereof through software programs including Common Gateway Interface (CGI) programs, plug-ins, servlets, active server pages, server side include (SSI) functions or the like.

A representative Web client is a personal computer that is x86-, PowerPC®-or RISC-based, that includes an operating system such as IBM® OS/2® or Microsoft Windows '95, and that includes a Web browser, such as Netscape Navigator 4.0 (or higher), having a Java Virtual Machine (JVM) and support for application plug-ins or helper applications. A client may also be a notebook computer, a handheld computing device (e.g., a PDA), an Internet appliance, or any other such device connectable to the computer network.

As seen in FIG. 2, a typical Web page comprises a markup language (e.g. HTML) master or base document 28, and many embedded objects (e.g., images, audio, video, or the like) 30. Thus, in a typical page, twenty or more embedded images or objects are quite common. Each of these images is an independent object in the Web, retrieved (or validated for change) separately. The common behavior of a Web client, therefore, is to fetch the base HTML document, and then immediately fetch the embedded objects, which are typically (but not always) located on the same server. According to the present invention, preferably the markup language base document 28 is served from the Web server (i.e., the Content Provider site) whereas a given number (or perhaps all) of the embedded objects are served from other servers. As will be seen, preferably a given embedded object is served from a server (other than the Web server itself) that is close to the client machine, that is not overloaded, and that is most likely to already have a current version of the required file.

Referring now to FIG. 3, this operation is achieved by the hosting system of the present invention. As will be seen, the hosting system 35 comprises a set of widely-deployed servers (or server resources) that form a large, fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users. The servers may be deployed globally, or across any desired geographic regions. As will be seen, the hosting system provides a distributed architecture for intelligently routing and replicating such content. To this end, the global hosting system 35 comprises three (3) basic types of servers (or server resources): hosting servers (sometimes called ghosts) 36, top-level DNS servers 38, and low-level DNS servers 40. Although not illustrated, there may be additional levels in the DNS hierarchy. Alternatively, there may be a single DNS level that combines the functionality of the top level and low-level servers. In this illustrative embodiment, the inventive framework 35 is deployed by an Internet Service Provider (ISP), although this is not a limitation of the present invention. The ISP or ISPs that deploy the inventive global hosting framework 35 preferably have a large number of machines that run both the ghost server component 36 and the low-level DNS component 40 on their networks. These machines are distributed throughout the network; preferably, they are concentrated around network exchange points 42 and network access points 44, although this is not a requirement. In addition, the

ISP preferably has a small number of machines running the top-level DNS 38 that may also be distributed throughout the network.

Although not meant to be limiting, preferably a given server used in the framework 35 includes a processor, an operating system (e.g., Linux, UNIX, Windows NT, or the like), a Web server application, and a set of application routines used by the invention. These routines are conveniently implemented in software as a set of instructions executed by the processor to perform various process or method steps as will be described in more detail below. The servers are preferably located at the edges of the network (e.g., in points of presence, or POPs).

Several factors may determine where the hosting servers are placed in the network. Thus, for example, the server locations are preferably determined by a demand driven network map that allows the provider (e.g., the ISP) to monitor traffic requests. By studying traffic patterns, the ISP may optimize the server locations for the given traffic profiles.

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. Thus, preferably, the base HTML document is served from the Content Provider that normally hosts the page. The embedded objects, or some subset thereof, are preferentially served from the hosting servers 36 and, specifically, given hosting servers 36 that are near the client machine that in the first instance initiated the request for the Web page. In addition, preferably loads across the hosting servers are balanced to ensure that a given embedded object may be efficiently served from a given hosting server near the client when such client requires that object to complete the page.

To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well-known, each embedded object that may be served in a page has its own URL. Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served, i.e., without reference to the present invention. According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers. A flowchart illustrating the preferred method for modifying the object URL is illustrated in FIG. 4.

The routine begins at step 50 by determining whether all of the embedded objects in a given page have been processed. If so, the routine ends. If not, however, the routine gets the next embedded object at step 52. At step 54, a virtual server hostname is prepended into the URL for the given embedded object. The virtual server hostname includes a value (e.g., a number) that is generated, for example, by applying a given hash function to the URL. As is well-known, a hash function takes arbitrary length bit strings as inputs and produces fixed length bit strings (hash values) as outputs. Such functions satisfy two conditions: (1) it is infeasible to find two different inputs that produce the same hash value, and (2) given an input and its hash value, it is infeasible to find a different input with the same hash value. In step 54, the URL for the embedded object is hashed into a value xx,xxx that is then included in the virtual server hostname. This step randomly distributes the object to a given virtual server hostname.

The present invention is not limited to generating the virtual server hostname by applying a hash function as described above. As an alternative and preferred

embodiment, a virtual server hostname is generated as follows. Consider the representative hostname a1234.g.akamai.tech.net. The 1234 value, sometimes referred to as a serial number, preferably includes information about the object such as its size (big or small), its anticipated popularity, the date on which the object was created, the identity of the Web site, the type of object (e.g., movie or static picture), and perhaps some random bits generated by a given random function. Of course, it is not required that any given serial number encode all of such information or even a significant number of such components. Indeed, in the simplest case, the serial number may be a simple integer. In any event, the information is encoded into a serial number in any convenient manner. Thus, for example, a first bit is used to denote size, a second bit is used to denote popularity, a set of additional bits is used to denote the date, and so forth. As noted above in the hashing example, the serial number is also used for load balancing and for directing certain types of traffic to certain types of servers. Typically, most URLs on the same page have the same serial number to minimize the number of distinguished name (DN) accesses needed per page. This requirement is less important for larger objects.

Thus, according to the present invention, a virtual server hostname is prepended into the URL for a given embedded object, and this hostname includes a value (or serial number) that is generated by applying a given function to the URL or object. That function may be a hash function, an encoding function, or the like.

Turning now back to the flowchart, the routine then continues at step 56 to include a given value in the object's URL. Preferably, the given value is generated by applying a given hash function to the embedded object. This step creates a unique fingerprint of the object that is useful for determining whether the object has been modified. Thereafter, the routine returns to step 50 and cycles.

With the above as background, the inventive global hosting framework is now described in the context of a specific example. In particular, it is assumed that a user of a client machine in Boston requests a Content Provider Web page normally hosted in Atlanta. For illustrative purposes, it is assumed that the Content Provider is using the global hosting architecture within a network, which may be global, international, national, regional, local or private. FIG. 5 shows the various components of the system and how the request from the client is processed. This operation is not to be taken by way of limitation, as will be explained.

Step 1: The browser sends a request to the Provider's Web site (Item 1). The Content Provider site in Atlanta receives the request in the same way that it does as if the global hosting framework were not being implemented. The difference is in what is returned by the Provider site. Instead of returning the usual page, according to the invention, the Web site returns a page with embedded object URLs that are modified according to the method illustrated in the flowchart of FIG. 4. As previously described, the URLs preferably are changed as follows:

Assume that there are 100,000 virtual ghost servers, even though there may only be a relatively small number (e.g., 100) physically present on the network. These virtual ghost servers or virtual ghosts are identified by the hostname: ghostxxxxx.ghosting.com, where xxxxx is replaced by a number between 0 and 99,999. After the Content Provider Web site is updated with new information, a script executing on the Content Provider site is run that rewrites the embedded URLs. Preferably, the embedded URLs names are

hashed into numbers between 0 and 99,999, although this range is not a limitation of the present invention. An embedded URL is then switched to reference the virtual ghost with that number. For example, the following is an embedded URL from the Provider's site:

If the serial number for the object referred to by this URL is the number 1467, then preferably the URL is rewritten to read:

The use of serial numbers in this manner distributes the embedded URLs roughly evenly over the 100,000 virtual ghost server names. Note that the Provider site can still personalize the page by rearranging the various objects on the screen according to individual preferences. Moreover, the Provider can also insert advertisements dynamically and count how many people view each ad.

According to the preferred embodiment, an additional modification to the embedded URLs is made to ensure that the global hosting system does not serve stale information. As previously described, preferably a hash of the data contained in the embedded URL is also inserted into the embedded URL itself. That is, each embedded URL may contain a fingerprint of the data to which it points. When the underlying information changes, so does the fingerprint, and this prevents users from referencing old data.

The second hash takes as input a stream of bits and outputs what is sometimes referred to as a fingerprint of the stream. The important property of the fingerprint is that two different streams almost surely produce two different fingerprints. Examples of such hashes are the MD2 and MD5 hash functions, however, other more transparent methods such as a simple checksum may be used. For concreteness, assume that the output of the hash is a 128 bit signature. This signature can be interpreted as a number and then inserted into the embedded URL. For example, if the hash of the data in the picture space.story.gif from the Provider web site is the number 28765, then the modified embedded URL would actually look as follows:

Whenever a page is changed, preferably the hash for each embedded URL is recomputed and the URL is rewritten if necessary. If any of the URL's data changes, for example, a new and different picture is inserted with the name space.story.gif, then the hash of the data is different and therefore the URL itself will be different. This scheme prevents the system from serving data that is stale as a result of updates to the original page.

For example, assume that the picture space.story.gif is replaced with a more up-to-date version on the Content Provider server. Because the data of the pictures changes, the hash of the URL changes as well. Thus, the new embedded URL looks the same except that a new number is inserted for the fingerprint. Any user that requests the page after the update receives a page that points to the new picture. The old picture is never referenced and cannot be mistakenly returned in place of the more up-to-date information.

In summary, preferably there are two hashing operations that are done to modify the pages of the Content Provider. First, hashing can be a component of the process by which a serial number is selected to transform the domain name into a virtual ghost name. As will be seen, this first transformation serves to redirect clients to the global hosting

system to retrieve the embedded URLs. Next, a hash of the data pointed to by the embedded URLs is computed and inserted into the URL. This second transformation serves to protect against serving stale and out-of-date content from the ghost servers. Preferably, these two transformations are performed off-line and therefore do not pose potential performance bottlenecks.

Generalizing, the preferred URL schema is as follows. The illustrative domain `www.domainname.com/frontpage.jpg` is transformed into:

```
xxxx.yy.zzzz.net/aaaa/www.domainname.com/
frontpage.jpg,
where:
xxxx=serial number field
yy=lower level DNS field
zzzz=top level DNS field
aaaa=other information (e.g., fingerprint) field.
```

If additional levels of the DNS hierarchy are used, then there may be additional lower level DNS fields, e.g., `xxxx.y1y1.y2y2.zzz.net/aaaa/. . .`

Step 2: After receiving the initial page from the Content Provider site, the browser needs to load the embedded URLs to display the page. The first step in doing this is to contact the DNS server on the user's machine (or at the user's ISP) to resolve the altered hostname, in this case: `ghost1467.ghosting.akamai.com`. As will be seen, the global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. To appreciate how this is done, the following describes the progress of the DNS query that was initiated by the client.

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for `ghosting.com` have a special function that is different from regular DNS servers like those of the `.com` domain. The top level DNS servers 38 include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to a `akamai.com` (i.e., a low level DNS) server 40 that is close-by. Like the `.com` domain, `akamai.com` preferably has a number of top-level DNS servers 38 spread throughout the network for fault tolerance. Thus, a given top level DNS server 38 directs the user to a region in the Internet (having a collection of hosting servers 36 that may be used to satisfy the request for a given embedded object) whereas the low level DNS server 40 (within the identified region) identifies a particular hosting server within that collection from which the object is actually served.

More generally, as noted above, the DNS process can contain several levels of processing, each of which serves to better direct the client to a ghost server. The ghost server name can also have more fields. For example, "`a123.g.g.akamaitech.net`" may be used instead of "`a123.ghost.akamai.com`." If only one DNS level is used, a representative URL could be "`a123.akamai.com`."

Although other techniques may be used, the user's location in the network preferably is deduced by looking at the IP address of the client machine making the request. In the present example, the DNS server is running on the machine of the user, although this is not a requirement. If the user is using an ISP DNS server, for example, the routines make the assumption that the user is located near (in the Internet sense) this server. Alternatively, the user's location or IP address could be directly encoded into the request sent to the top level DNS. To determine the physical location of an IP address in the network, preferably, the top level DNS server builds a network map that is then used to identify the relevant location.

Thus, for example, when a request comes in to a top level DNS for a resolution for `a1234.g.akamaitech.net`, the top level DNS looks at the return address of the requester and then formulates the response based on that address according to a network map. In this example, the `a1234` is a serial number, the `g` is a field that refers to the lower level DNS, and `akamaitech` refers to the top level DNS. The network map preferably contains a list of all Internet Protocol (IP) blocks and, for each IP block, the map determines where to direct the request. The map preferably is updated continually based on network conditions and traffic.

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network. The ability to redirect requests is a standard feature in the DNS system. In addition, this redirection can be done in such a way that if the local low level DNS server is down, there is a backup server that is contacted.

Preferably, the TTL (time to live) stamp on these top level DNS redirections for the `ghosting.com` domain is set to be long. This allows DNS caching at the user's DNS servers and/or the ISP's DNS servers to prevent the top level DNS servers from being overloaded. If the TTL for `ghosting.akamai.com` in the DNS server at the user's machine or ISP has expired, then a top level server is contacted, and a new redirection to a local low level `ghosting.akamai.com` DNS server is returned with a new TTL stamp. It should be noted the system does not cause a substantially larger number of top level DNS lookups than what is done in the current centralized hosting solutions. This is because the TTL of the top level redirections are set to be high and, thus, the vast majority of users are directed by their local DNS straight to a nearby low level `ghosting.akamai.com` DNS server.

Moreover, fault tolerance for the top level DNS servers is provided automatically by DNS similarly to what is done for the popular `.com` domain. Fault tolerance for the low level DNS servers preferably is provided by returning a list of possible low level DNS servers instead of just a single server. If one of the low level DNS servers is down, the user will still be able to contact one on the list that is up and running.

Fault tolerance can also be handled via an "overflow control" mechanism wherein the client is redirected to a low-level DNS in a region that is known to have sufficient capacity to serve the object. This alternate approach is very useful in scenarios where there is a large amount of demand from a specific region or when there is reduced capacity in a region. In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded. Minimizing overall latency subject to the regional capacity constraints preferably is achieved using a min-cost multicommodity flow algorithm.

Step 4: At this point, the user has the address of a close-by `ghosting.com` DNS server 38. The user's local DNS server contacts the close-by low level DNS server 40 and requests a translation for the name `ghost1467.ghosting.akamai.com`. The local DNS server is responsible for returning the IP address of one of the ghost servers 36 on the network that is close to the user, not overloaded, and most likely to already have the required data.

The basic mechanism for mapping the virtual ghost names to real ghosts is hashing. One preferred technique is so-called consistent hashing, as described in U.S. Ser. No. 09/042,228, filed Mar. 13, 1998, and in U.S. Ser. No. 09/088,825, filed Jun. 2, 1998, each titled Method And Apparatus For Distributing Requests Among A Plurality Of

Resources, and owned by the Massachusetts Institute of Technology, which applications are incorporated herein by reference. Consistent hash functions make the system robust under machine failures and crashes. It also allows the system to grow gracefully, without changing where most items are located and without perfect information about the system.

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance. Preferably, a table of resolutions is created for each serial number. For example, serial number 1 resolves to ghost 2 and 5, serial number 2 resolves to ghost 3, serial number 3 resolves to ghosts 2,3,4, and so forth. The goal is to define the resolutions so that no ghost exceeds its capacity and that the total number of all ghosts in all resolutions is minimized. This is done to assure that the system can take maximal advantage of the available memory at each region. This is a major advantage over existing load balancing schemes that tend to cache everything everywhere or that only cache certain objects in certain locations no matter what the loads are. In general, it is desirable to make assignments so that resolutions tend to stay consistent over time provided that the loads do not change too much in a short period of time. This mechanism preferably also takes into account how close the ghost is to the user, and how heavily loaded the ghost is at the moment.

Note that the same virtual ghost preferably is translated to different real ghost addresses according to where the user is located in the network. For example, assume that ghost server 18.98.0.17 is located in the United States and that ghost server 132.68.1.28 is located in Israel. A DNS request for ghost1487.ghosting.akamai.com originating in Boston will resolve to 18.98.0.17, while a request originating in Tel-Aviv will resolve to 132.68.1.28.

The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers. In one embodiment, the load information is circulated among the servers in a region so that they can compute resolutions for each serial number. One algorithm for computing resolutions works as follows. The server first computes the projected load (based on number of user requests) for each serial number. The serial numbers are then processed in increasing order of load. For each serial number, a random priority list of desired servers is assigned using a consistent hashing method. Each serial number is then resolved to the smallest initial segment of servers from the priority list so that no server becomes overloaded. For example, if the priority list for a serial number is 2,5,3,1,6, then an attempt is made first to try to map the load for the serial number to ghost 2. If this overloads ghost 2, then the load is assigned to both ghosts 2 and 5. If this produced too much load on either of those servers, then the load is assigned to ghosts 2,3, and 5, and so forth. The projected load on a server can be computed by looking at all resolutions that contain that server and by adding the amount of load that is likely to be sent to that server from that serial number. This method of producing resolutions is most effective when used in an iterative fashion, wherein the assignments starts in a default state, where every serial number is mapped to every ghost. By refining the resolution table according to the previous procedure, the load is balanced using the minimum amount of replication (thereby maximally conserving the available memory in a region).

The TTL for these low level DNS translations is set to be short to allow a quick response when heavy load is detected

on one of the ghosts. The TTL is a parameter that can be manipulated by the system to insure a balance between timely response to high load on ghosts and the load induced on the low level DNS servers. Note, however, that even if the TTL for the low level DNS translation is set to 1–2 minutes, only a few of the users actually have to do a low level DNS lookup. Most users will see a DNS translation that is cached on their machine or at their ISP. Thus, most users go directly from their local DNS server to the close-by ghost that has the data they want. Those users that actually do a low level DNS lookup have a very small added latency, however this latency is small compared to the advantage of retrieving most of the data from close by.

As noted above, fault tolerance for the low level DNS servers is provided by having the top level DNS return a list of possible low level DNS servers instead of a single server address. The user's DNS system caches this list (part of the standard DNS system), and contacts one of the other servers on the list if the first one is down for some reason. The low level DNS servers make use of a standard feature of DNS to provide an extra level of fault tolerance for the ghost servers. When a name is translated, instead of returning a single name, a list of names is returned. If for some reason the primary fault tolerance method for the ghosts (known as the Buddy system, which is described below) fails, the client browser will contact one of the other ghosts on the list.

Step 5: The browser then makes a request for an object named a123.ghosting.akamai.com/.../www.provider.com/TECH/images/space.story.gif from the close-by ghost. Note that the name of the original server (www.provider.com) preferably is included as part of the URL. The software running on the ghost parses the page name into the original host name and the real page name. If a copy of the file is already stored on the ghost, then the data is returned immediately. If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server or another ghost server. Note that the ghost knows who the original server was because the name was encoded into the URL that was passed to the ghost from the browser. Once a copy has been retrieved it is returned to the user, and preferably it is also stored on the ghost for answering future requests.

As an additional safeguard, it may be preferable to check that the user is indeed close to the server. This can be done by examining the IP address of the client before responding to the request for the file. This is useful in the rare case when the client's DNS server is far away from the client. In such a case, the ghost server can redirect the user to a closer server (or to another virtual address that is likely to be resolved to a server that is closer to the client). If the redirect is to a virtual server, then it must be tagged to prevent further redirections from taking place. In the preferred embodiment, redirection would only be done for large objects; thus, a check may be made before applying a redirection to be sure that the object being requested exceeds a certain overall size.

Performance for long downloads can also be improved by dynamically changing the server to which a client is connected based on changing network conditions. This is especially helpful for audio and video downloads (where the connections can be long and where quality is especially important). In such cases, the user can be directed to an alternate server in mid-stream. The control structure for redirecting the client can be similar to that described above, but it can also include software that is placed in the client's browser or media player. The software monitors the performance of the client's connection and perhaps the status of the network as well. If it is deemed that the client's connection can be improved by changing the server, then the system directs the client to a new server for the rest of the connection.

Fault tolerance for the ghosts is provided by a buddy system, where each ghost has a designated buddy ghost. If a ghost goes down, its buddy takes over its work (and IP address) so that service is not interrupted. Another feature of the system is that the buddy ghost does not have to sit idle waiting for a failure. Instead, all of the machines are always active, and when a failure happens, the load is taken over by the buddy and then balanced by the low level DNS system to the other active ghosts. An additional feature of the buddy system is that fault tolerance is provided without having to wait for long timeout periods.

As yet another safety feature of the global hosting system, a gating mechanism can be used to keep the overall traffic for certain objects within specified limits. One embodiment of the gating mechanism works as follows. When the number of requests for an object exceeds a certain specified threshold, then the server can elect to not serve the object. This can be very useful if the object is very large. Instead, the client can be served a much smaller object that asks the client to return later. Or, the client can be redirected. Another method of implementing a gate is to provide the client with a "ticket" that allows the client to receive the object at a prespecified future time. In this method, the ghost server needs to check the time on the ticket before serving the object.

The inventive global hosting scheme is a way for global ISPs or conglomerates of regional ISPs to leverage their network infrastructure to generate hosting revenue, and to save on network bandwidth. An ISP offering the inventive global hosting scheme can give content providers the ability to distribute content to their users from the closest point on the ISPs network, thus ensuring fast and reliable access. Guaranteed web site performance is critical for any web-based business, and global hosting allows for the creation of a service that satisfies this need.

Global hosting according to the present invention also allows an ISP to control how and where content traverses its network. Global hosting servers can be set up at the edges of the ISP's network (at the many network exchange and access points, for example). This enables the ISP to serve content for sites that it hosts directly into the network exchange points and access points. Expensive backbone links no longer have to carry redundant traffic from the content provider's site to the network exchange and access points. Instead, the content is served directly out of the ISP's network, freeing valuable network resources for other traffic.

Although global hosting reduces network traffic, it is also a method by which global ISPs may capture a piece of the rapidly expanding hosting market, which is currently estimated at over a billion dollars a year.

The global hosting solution also provides numerous advantages to Content Providers, and, in particular, an efficient and cost-effective solution to improve the performance of their Web sites both domestically and internationally. The inventive hosting software ensures Content Providers with fast and reliable Internet access by providing a means to distribute content to their subscribers from the closest point on an ISP's network. In addition to other benefits described in more detail below, the global hosting solution also provides the important benefit of reducing network traffic.

Once inexpensive global hosting servers are installed at the periphery of an ISP's network (i.e., at the many network exchange and access points), content is served directly into network exchange and access points. As a result of this efficient distribution of content directly from an ISP's network, the present invention substantially improves Web

site performance. In contrast to current content distribution systems, the inventive global hosting solution does not require expensive backbone links to carry redundant traffic from the Content Provider's Web site to the network exchange and access points.

A summary of the specific advantages afforded by the inventive global hosting scheme are set forth below:

1. Decreased Operational Expenses for Content Providers:

Most competing solutions require Content Providers to purchase servers at each Web site that hosts their content. As a result, Content Providers often must negotiate separate contracts with different ISPs around the world. In addition, Content Providers are generally responsible for replicating the content and maintaining servers in these remote locations.

With the present invention, ISPs are primarily responsible for the majority of the aspects of the global hosting. Content Providers preferably maintain only their single source server. Content on this server is automatically replicated by software to the locations where it is being accessed. No intervention or planning is needed by the Provider (or, for that matter, the ISP). Content Providers are offered instant access to all of the servers on the global network; there is no need to choose where content should be replicated or to purchase additional servers in remote locations.

2. Intelligent and Efficient Data Replication:

Most competing solutions require Content Providers to replicate their content on servers at a commercial hosting site or to mirror their content on geographically distant servers. Neither approach is particularly efficient. In the former situation, content is still located at a single location on the Internet (and thus it is far away from most users). In the latter case, the entire content of a Web site is copied to remote servers, even though only a small portion of the content may actually need to be located remotely. Even with inexpensive memory, the excessive cost associated with such mirroring makes it uneconomical to mirror to more than a few sites, which means that most users will still be far away from a mirror site. Mirroring also has the added disadvantage that Content Providers must insure that all sites remain consistent and current, which is a nontrivial task for even a few sites.

With the present invention, content is automatically replicated to the global server network in an intelligent and efficient fashion. Content is replicated in only those locations where it is needed. Moreover, when the content changes, new copies preferably are replicated automatically throughout the network.

3. Automatic Content Management:

Many existing solutions require active management of content distribution, content replication and load balancing between different servers. In particular, decisions about where content will be hosted must be made manually, and the process of replicating data is handled in a centralized push fashion. On the contrary, the invention features passive management. Replication is done in a demand-based pull fashion so that content preferably is only sent to where it is truly needed. Moreover, the process preferably is fully automated; the ISP does not have to worry about how and where content is replicated and/or the content provider.

4. Unlimited, Cost Effective Scalability:

Competing solutions are not scalable to more than a small number of sites. For example, solutions based on mirroring are typically used in connection with at most three or four sites. The barriers to scaling include the expense of replicating the entire site, the cost of replicating computing

resources at all nodes, and the complexity of supporting the widely varying software packages that Content Providers use on their servers.

The unique system architecture of the present invention is scalable to hundreds, thousands or even millions of nodes. Servers in the hosting network can malfunction or crash and the system's overall function is not affected. The global hosting framework makes efficient use of resources; servers and client software do not need to be replicated at every node because only the hosting server runs at each node. In addition, the global hosting server is designed to run on standard simple hardware that is not required to be highly fault tolerant.

5. Protection against Flash Crowds:

Competing solutions do not provide the Content Provider with protection from unexpected flash crowds. Although mirroring and related load-balancing solutions do allow a Content Provider to distribute load across a collection of servers, the aggregate capacity of the servers must be sufficient to handle peak demands. This means that the Provider must purchase and maintain a level of resources commensurate with the anticipated peak load instead of the true average load. Given the highly variable and unpredictable nature of the Internet, such solutions are expensive and highly wasteful of resources.

The inventive hosting architecture allows ISPs to utilize a single network of hosting servers to offer Content Providers flash crowd insurance. That is, insurance that the network will automatically adapt to and support unexpected higher load on the Provider's site. Because the ISP is aggregating many Providers together on the same global network, resources are more efficiently used.

6. Substantial Bandwidth Savings:

Competing solutions do not afford substantial bandwidth savings to ISPs or Content Providers. Through the use of mirroring, it is possible to save bandwidth over certain links (i.e., between New York and Los Angeles). Without global hosting, however, most requests for content will still need to transit the Internet, thus incurring bandwidth costs. The inventive hosting framework saves substantial backbone bandwidth for ISPs that have their own backbones. Because content is distributed throughout the network and can be placed next to network exchange points, both ISPs and Content Providers experience substantial savings because backbone charges are not incurred for most content requests.

7. Instant Access to the Global Network:

Competing solutions require the Content Provider to choose manually a small collection of sites at which content will be hosted and/or replicated. Even if the ISP has numerous hosting sites in widely varied locations, only those sites specifically chosen (and paid for) will be used to host content for that Content Provider.

On the contrary, the global hosting solution of the present invention allows ISPs to offer their clients instant access to the global network of servers. To provide instant access to the global network, content is preferably constantly and dynamically moved around the network. For example, if a Content Provider adds content that will be of interest to customers located in Asia, the Content Provider will be assured that its content will be automatically moved to servers that are also located in Asia. In addition, the global hosting framework allows the content to be moved very close to end users (even as close as the user's building in the case of the Enterprise market).

8. Designed for Global ISPs and Conglomerates:

Most competing solutions are designed to be purchased and managed by Content Providers, many of whom are

already consistently challenged and consumed by the administrative and operational tasks of managing a single server. The inventive hosting scheme may be deployed by a global ISP, and it provides a new service that can be offered to Content Providers. A feature of the service is that it minimizes the operational and managerial requirements of a Content Provider, thus allowing the Content Provider to focus on its core business of creating unique content.

9. Effective Control of Proprietary Database s a nd Confidential Information:

Many competing solutions require Content Providers to replicate their proprietary databases to multiple geographically distant sites. As a result, the Content Provider effectively loses control over its proprietary and usually confidential databases. To remedy these problems, the global hosting solution of the present invention ensures that Content Providers retain complete control over their databases. As described above, initial requests for content are directed to the Content Provider's central Web site, which then implements effective a nd controlled database access. Preferably, high-bandwidth, static parts for page requests are retrieved from the global hosting network.

10. Compatibility with Content Provider Software:

Many competing solutions require Content Provider s to utilize a specific set of servers and databases. These particular, non-uniform requirements constrain the Content Provider's ability to most effectively use new technologies, and may require expensive changes to a Content Provider's existing infrastructure. By eliminating these problems, the inventive global hosting architecture effectively interfaces between the Content Provider and the ISP, and it does not make any assumptions about the systems or servers used by the Content Provider. Furthermore, the Content Provider's systems can be upgraded, changed or completely replaced without modifying or interrupting the inventive architecture.

11. No Interference with Dynamic Content, Personalized Advertising or E-Commerce, and No stale content:

Many competing solutions (such as naive caching of all content) can interfere with dynamic content, personalized advertising and E-commerce and can serve the user with stale content. While other software companies have attempted to partially eliminate these issues (such as keeping counts on hits for all cached copies), each of these solutions causes a partial or complete loss of functionality (such as the ability to personalize advertising). On the contrary, the global hosting solution does not interfere with generation of dynamic content, personalized advertising or E-commerce, because each of these tasks preferably is handled by the central server of the Content Provider.

12. Designed for the Global Network:

The global hosting architecture is highly scalable and thus may be deployed on a world-wide network basis.

The above-described functionality of each of the components of the global hosting architecture preferably is implemented in software executable in a processor, namely, as a set of instructions or program code in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network.

In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such

methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Further, as used herein, a Web “client” should be broadly construed to mean any computer or component thereof directly or indirectly connected or connectable in any known or later-developed manner to a computer network, such as the Internet. The term Web “server” should also be broadly construed to mean a computer, computer platform, an adjunct to a computer or platform, or any component thereof. Of course, a “client” should be broadly construed to mean one who requests or gets the file, and “server” is the entity which downloads the file.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is set forth in the following claims:

1. A distributed hosting framework operative in a computer network in which users of client machines connect to a content provider server, the framework comprising:

a routine for modifying at least one embedded object URL of a web page to include a hostname pretended to a domain name and path;

a set of content servers, distinct from the content provider server, for hosting at least some of the embedded objects of web pages that are normally hosted by the content provider server;

at least one first level name server that provides a first level domain name service (DNS) resolution; and

at least one second level name server that provides a second level domain name service (DNS) resolution;

wherein in response to requests for the web page, generated by the client machines the web page including the modified embedded object URL is served from the content provider server and the embedded object identified by the modified embedded object URL is served from a given one of the content servers as identified by the first level and second level name servers.

2. The hosting framework as described in claim 1 further including a redundant first level name server.

3. The hosting framework as described in claim 1 further including a redundant second level name server.

4. The hosting framework as described in claim 1 wherein a given one of the set of servers includes a buddy server for assuming the hosting responsibilities of the given one of the set of servers upon a given failure condition.

5. The hosting framework as described in claim 1 wherein the second level name server includes a load balancing mechanism that balances loads across a subset of the set of servers.

6. The hosting framework as described in claim 5 wherein the load balancing mechanism minimizes the amount of replication required for the embedded objects while not exceeding a capacity of any of the set of servers.

7. The hosting framework as described in claim 1 further including an overflow control mechanism for minimizing an overall amount of latency experienced by client machines while not exceeding the capacity of any given subset of the set of servers.

8. The hosting framework as described in claim 7 wherein the overflow control mechanism includes a min-cost multi-commodity flow algorithm.

9. The hosting framework as described in claim 1 wherein the first level name server includes a network map for use in directing a request for the embedded object generated by a client.

10. The hosting framework as described in claim 1 wherein a server in the set of servers includes a gating

mechanism for maintaining overall traffic for a given embedded object within specified limits.

11. The hosting framework as described in claim 10 wherein the gating mechanism comprises:

means for determining whether a number of requests for the given embedded object exceeds a given threshold; and

means responsive to the determining means for restricting service of the given embedded object.

12. The hosting framework as described in claim 11 wherein the restricting means comprises means for serving an object that is smaller than the given embedded object.

13. The hosting framework as described in claim 11 wherein the object is a ticket that allows a client to receive the given embedded object at a later time.

14. A method of serving a page supported at a content provider server, the page comprising a markup language base document having associated therewith a set of embedded objects, each embedded object identified by a URL, comprising the steps of:

rewriting the URL of an embedded object to generate a modified URL, the modified URL including a new hostname prepended to an original hostname, wherein the original hostname is maintained as part of the modified URL for use in retrieving the embedded object whenever a cached copy of the embedded object is not available;

in response to a request to serve the page received at the content provider site, serving the page with the modified URL;

attempting to serve the embedded object from a content server other than the content provider server as identified by the new hostname; and

if the cached copy of the embedded object is not available from the content server, serving the embedded object from the content provider server.

15. A method of serving a page and an associated page object, wherein the page is stored on a content provider server and copies of the page object are stored on a set of content servers distinct from the content provider server, comprising the steps of:

(a) modifying a URL for the page object to include a hostname prepended to a content provider-supplied domain name and path;

(b) serving the page from the content provider server with the modified URL;

(c) responsive to a browser query to resolve the hostname, identifying a given one of the set of content servers from which the object may be retrieved; and

(d) returning to the browser an IP address of the identified content server to enable the browser to attempt to retrieve the object from that content server.

16. The method as described in claim 15 wherein the copies of the page object are stored on a subset of the set of content servers.

17. A content delivery method, comprising:

tagging an embedded object in a page to resolve to a domain other than a content provider domain by prepending given data to a content provider-supplied URL to generate an alternate resource locator (ARL); serving the page from a content provider server with the ARL; and

resolving the ARL to identify a content server in the domain; and

serving the embedded object from the identified content server.

19

18. The method as described in claim 17 wherein the step of resolving the ARL comprises:

utilizing a requesting user's location and data identifying then-current Internet traffic conditions to identify the content server.

19. A content delivery service, comprising:

replicating a set of page objects across a wide area network of content servers managed by a domain other than a content provider domain;

for a given page normally served from the content provider domain, tagging the embedded objects of the page so that requests for the page objects resolve to the domain instead of the content provider domain;

responsive to a request for the given page received at the content provider domain, serving the given page from the content provider domain; and

serving at least one embedded object of the given page from a given content server in the domain instead of from the content provider domain.

20. The content delivery method as described in claim 19 wherein the serving step comprises:

for each embedded object, identifying one or more content servers from which the embedded object may be retrieved.

21. The method as described in claim 20 wherein the identifying step comprises:

resolving a request to the domain as a function of a requesting user's location.

22. The method as described in claim 21 wherein the identifying step comprises:

resolving a request to the domain as a function of a requesting user's location and then-current Internet traffic conditions.

23. A method for Internet content delivery, comprising:

at the content provider server, modifying at least one embedded object URL of a page to include a hostname prepended to a domain name and a path normally used to retrieve the embedded object;

responsive to a request for the page issued from a client machine, serving the page with the modified embedded object URL to the client machine from the content provider server;

responsive to a request for the embedded object, resolving the hostname to an IP address of a content server, other than the content provider server, that is likely to host the embedded object; and

attempting to serve the embedded object to the client from the content server.

24. The method as described in claim 23 wherein the hostname includes a value generated by applying a given function to the embedded object.

20

25. The method as described in claim 24 wherein the value is generated by encoding given information, the given information selected from a group of information consisting essentially of: size data, popularity data, creation data and object type data.

26. The method as described in claim 4 wherein the given function randomly associates the embedded object with a virtual content bucket.

27. The method as described in claim 26 wherein the given function is an encoding function.

28. The method as described in claim 26 wherein the given function is a hash function.

29. The method as described in claim 23 wherein the modified URL also includes a fingerprint value generated by applying a given function to the embedded object.

30. The method as described in claim 29 wherein the value is a number generated by hashing the embedded object.

31. The method as described in claim 23 wherein the page is formatted according to a markup language.

32. The method as described in claim 23 further including the step of rewriting the embedded object URL as the content provider modifies the page.

33. The method as described in claim 23 wherein the step of resolving the hostname includes:

identifying a subset of content servers that may be available to serve the embedded object based on a location of the client machine and current Internet traffic conditions; and

identifying the content server from the subset of content servers.

34. A content delivery method, comprising:

distributing a set of page objects across a network of content servers managed by a domain other than a content provider domain, wherein the network of content servers are organized into a set of regions;

for a given page normally served from the content provider domain, tagging at least some of the embedded objects of the page so that requests for the objects resolve to the domain instead of the content provider domain;

in response to a client request for an embedded object of the page:

resolving the client request as a function of a location of the client machine making the request and current Internet traffic conditions to identify a given region; and

returning to the client an IP address of a given one of the content servers within the given region that is likely to host the embedded object and that is not overloaded.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,108,703

DATED : Aug. 22, 2000


INVENTOR(S) : F. Thomson Leighton, Daniel M. Lewin

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 11, line 1, delete "assachusetts" and substitute -- Massachusetts --.
In Column 11, line 2, delete "ncorporated" and substitute -- incorporated --.
In Column 11, line 3, delete "ake" and substitute -- make --.
In Column 16, line 9, delete "Database s a nd" and substitute -- Databases and --.
In Column 16, line 20, delete "a nd" and substitute -- and --.
In Column 16, line 24, delete "Provider s" and substitute -- Providers --.
In Claim 1, Column 17, line 21, delete "pretended" and substitute -- prepered --.
In Claim 1, Column 17, line 32, after "machines" insert -- , --.
In Claim 17, Column 18, line 64, delete "ARt" and substitute -- ARL --.

Signed and Sealed this
Fifteenth Day of May, 2001

Attest:



NICHOLAS P. GODICI

Attesting Officer

Acting Director of the United States Patent and Trademark Office

EXHIBIT H

United States Patent [19]

[11] **Patent Number:** **5,991,809**

Kriegsman

[45] **Date of Patent:** **Nov. 23, 1999**

- [54] **WEB SERVING SYSTEM THAT COORDINATES MULTIPLE SERVERS TO OPTIMIZE FILE TRANSFERS**

- | | | | |
|-----------|---------|----------------|------------|
| 5,715,453 | 2/1998 | Stewart | 395/615 |
| 5,721,914 | 2/1998 | DeVries | 395/200.31 |
| 5,734,831 | 3/1998 | Sanders | 395/200.53 |
| 5,742,762 | 4/1998 | Scholl et al. | 395/200.3 |
| 5,774,660 | 6/1998 | Brendel et al. | 395/200.31 |
| 5,796,952 | 8/1998 | Davis et al. | 395/200.54 |
| 5,828,847 | 10/1998 | Gehr et al. | 395/200.69 |

[75] Inventor: **Mark E. Kriegsman**, Boston, Mass.

[73] Assignee: **Clearway Technologies, LLC**, Boston, Mass.

[21] Appl. No.: 08/900,273

[22] Filed: **Jul. 25, 1997**

Related U.S. Application Data

- [60] Provisional application No. 60/022,598, Jul. 25, 1996.

- [51] **Int. Cl.**⁶ **G06F 13/00**

- [52] **U.S. Cl.** **709/226; 709/229**

- [58] **Field of Search** 395/200.31, 200.47,
395/200.48, 200.49, 200.59, 200.61, 200.62;
709/201, 226, 229, 231, 232, 219

[56] **References Cited**

U.S. PATENT DOCUMENTS

- | | | | |
|-----------|---------|---------------------|------------|
| 5,341,477 | 8/1994 | Pitkin et al. | 395/200.47 |
| 5,539,621 | 7/1996 | Kikinis | 361/803 |
| 5,572,643 | 11/1996 | Judson . | |
| 5,590,288 | 12/1996 | Castor et al. | 395/200.31 |
| 5,592,611 | 1/1997 | Midgely et al. | 395/182.02 |
| 5,619,648 | 4/1997 | Canale et al. . | |
| 5,623,656 | 4/1997 | Lyons . | |
| 5,625,781 | 4/1997 | Cline et al. . | |
| 5,649,186 | 7/1997 | Ferguson . | |
| 5,659,729 | 8/1997 | Nielsen . | |
| 5,666,362 | 9/1997 | Chen et al. . | |
| 5,671,279 | 9/1997 | Elgamal . | |

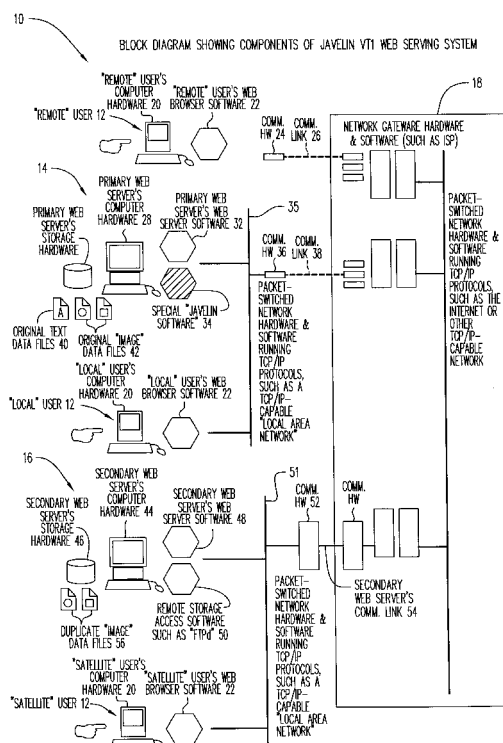
Primary Examiner—Zarni Maung

Attorney, Agent, or Firm—Ohlandt, Greeley, Ruggiero & Perle

[57] **ABSTRACT**

The present invention is a collaborative server system for providing high speed data transmission of data files across a communications network which, in brief summary, comprises a communications network, a primary server having a primary communications component for connecting the primary server to the communications network, and at least one secondary server having a secondary communications component for connecting the secondary server to the communications network. The primary server and the at least one secondary server include storage component for storing data files. The data files include static data files and/or dynamic data files. The storage component of the primary server further stores at least one look-up table having specific criteria pertaining to the data files and the primary and at least one secondary servers. The processor component of the primary server is effective to receive a request for specific data files from a network user, to look-up specific criteria in the look-up table pertaining to the specific data files, and to allocate transmission of each specific data file between the primary server and the at least one secondary server based on the specific criteria.

38 Claims, 7 Drawing Sheets



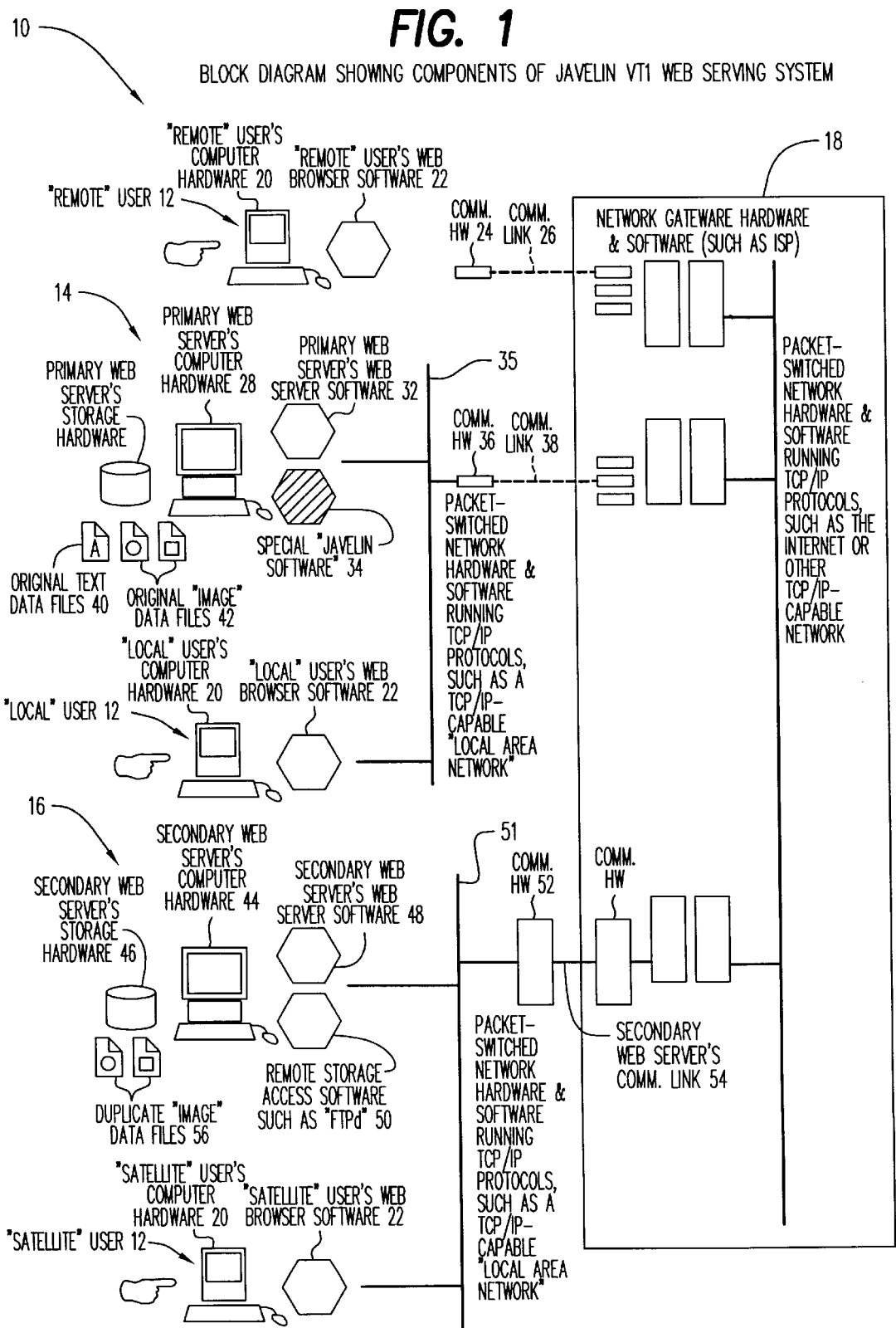


FIG. 2

MAIN FLOW CHART FOR SOFTWARE FOR JAVELIN VT1 WEB SERVING SYSTEM

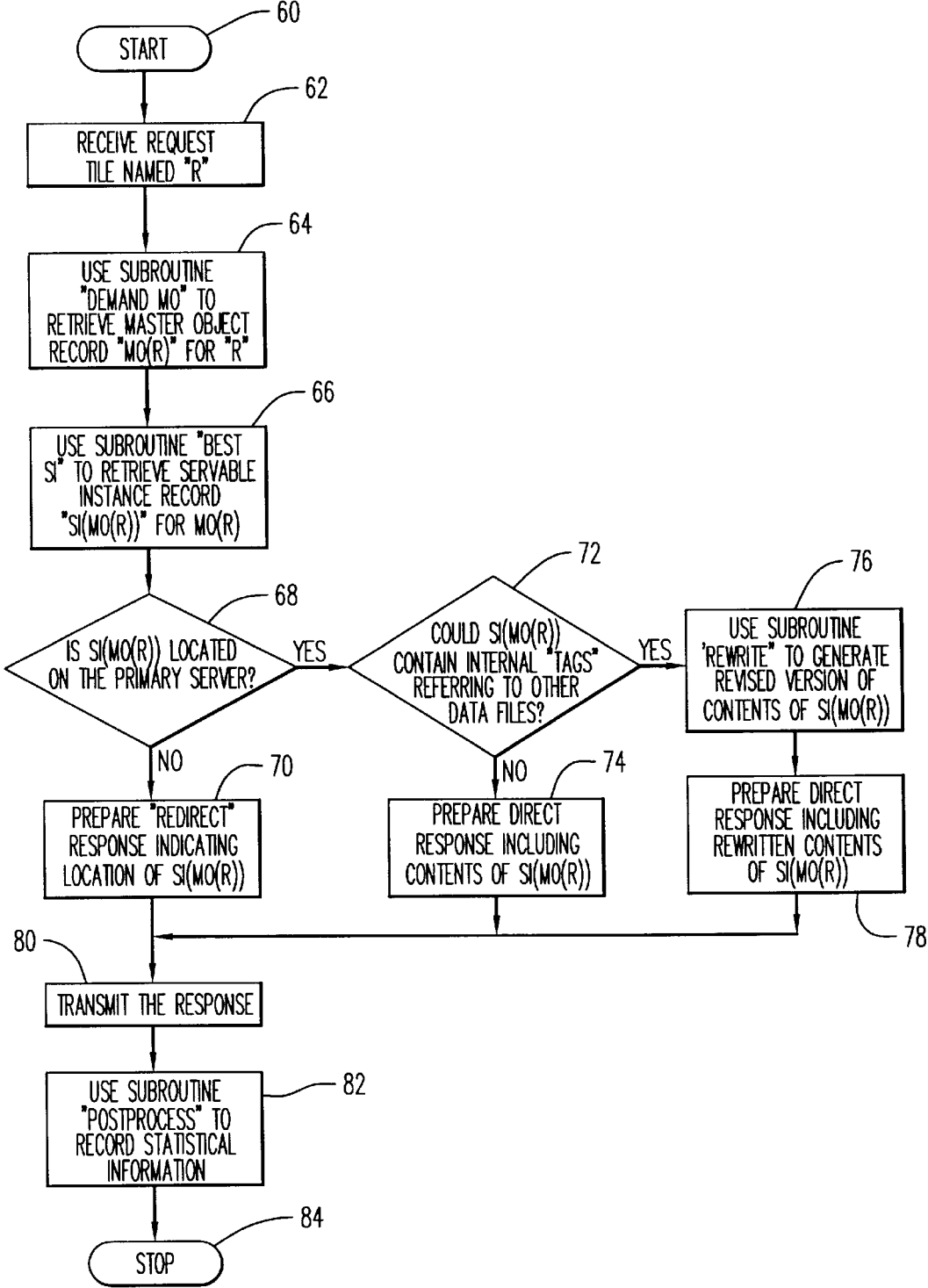


FIG. 3

SUBROUTINE "DEMAND MO"

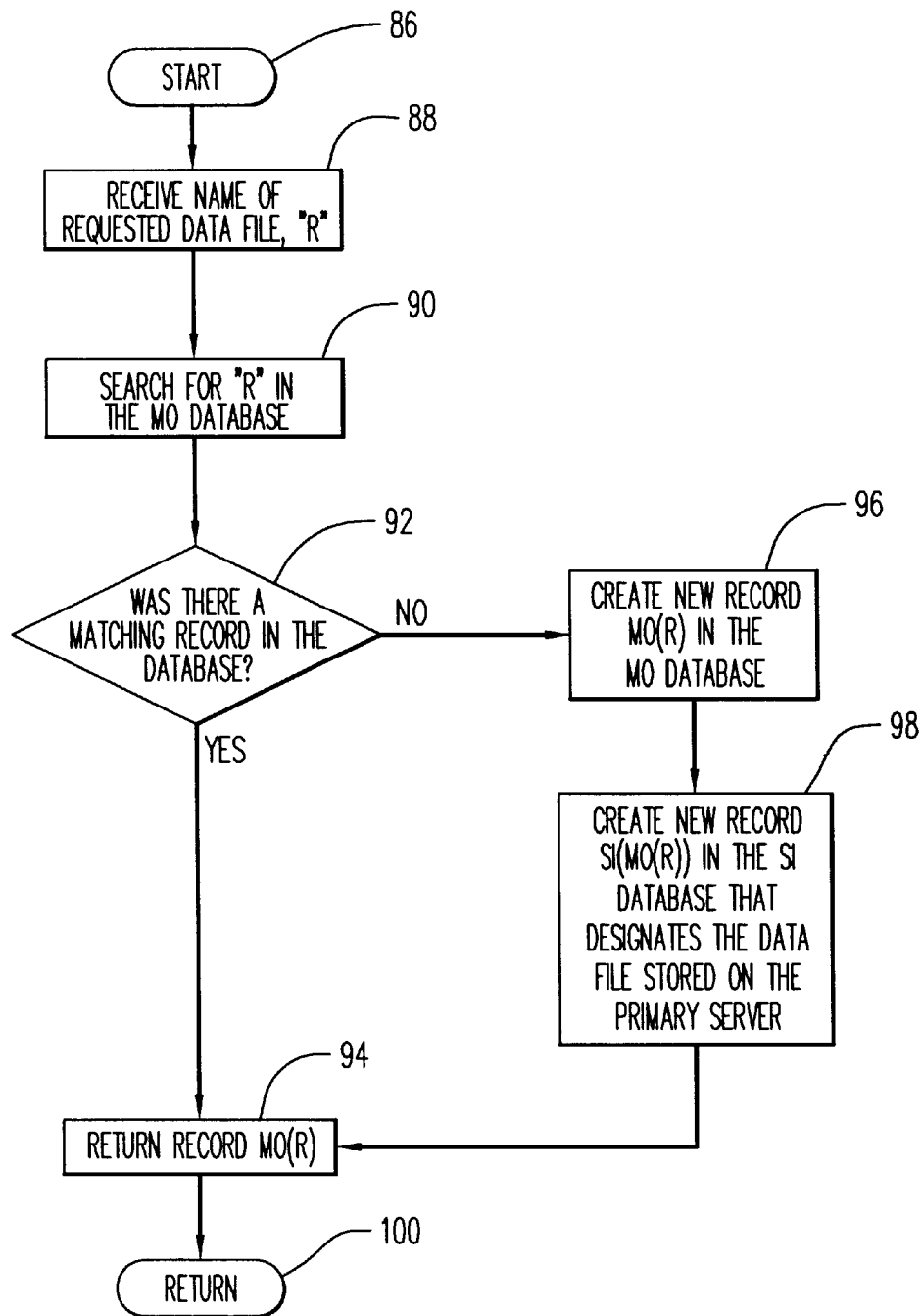


FIG. 4

SUBROUTINE "BEST SI"

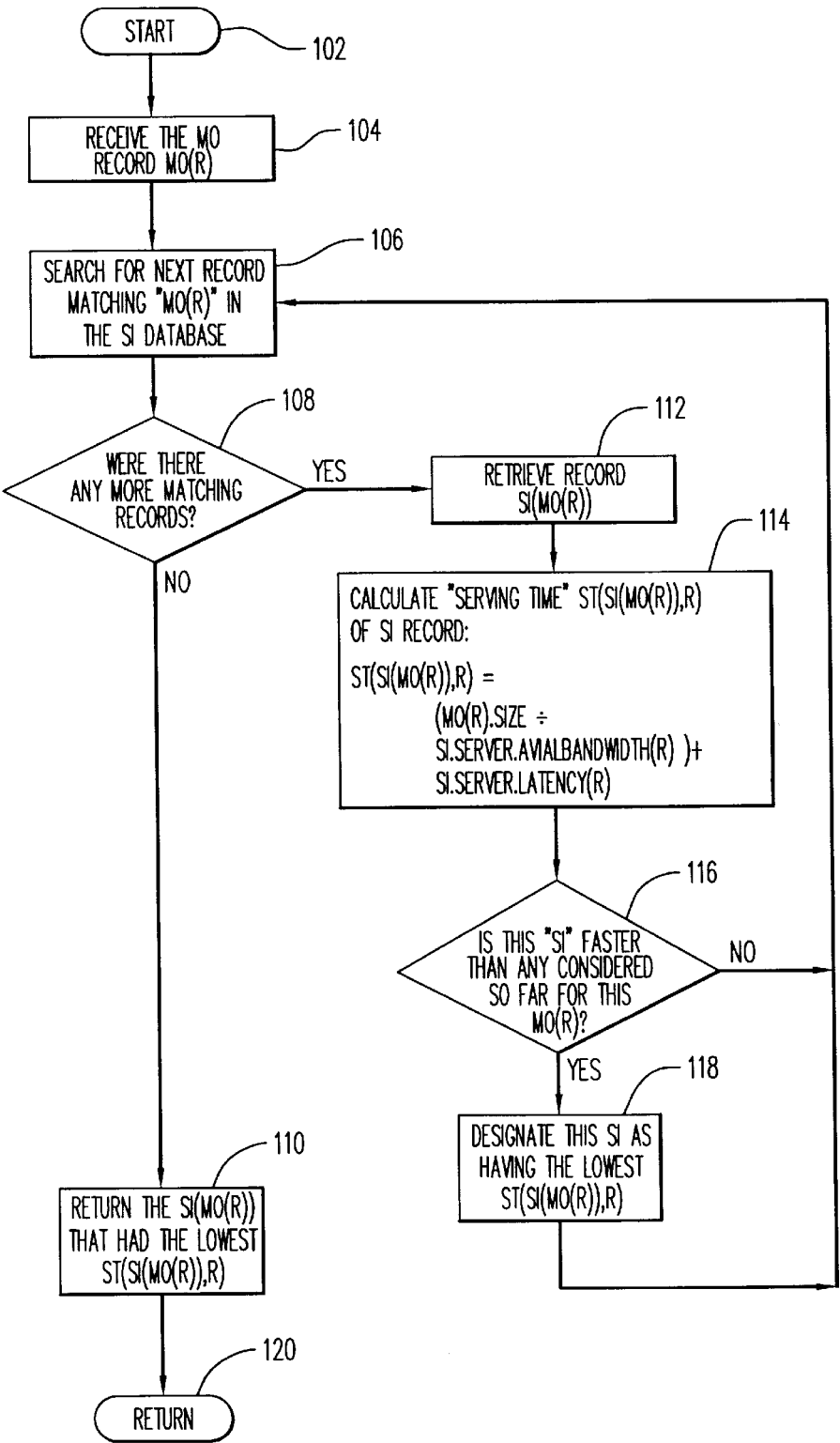


FIG. 5

SUBROUTINE "REWRITE"

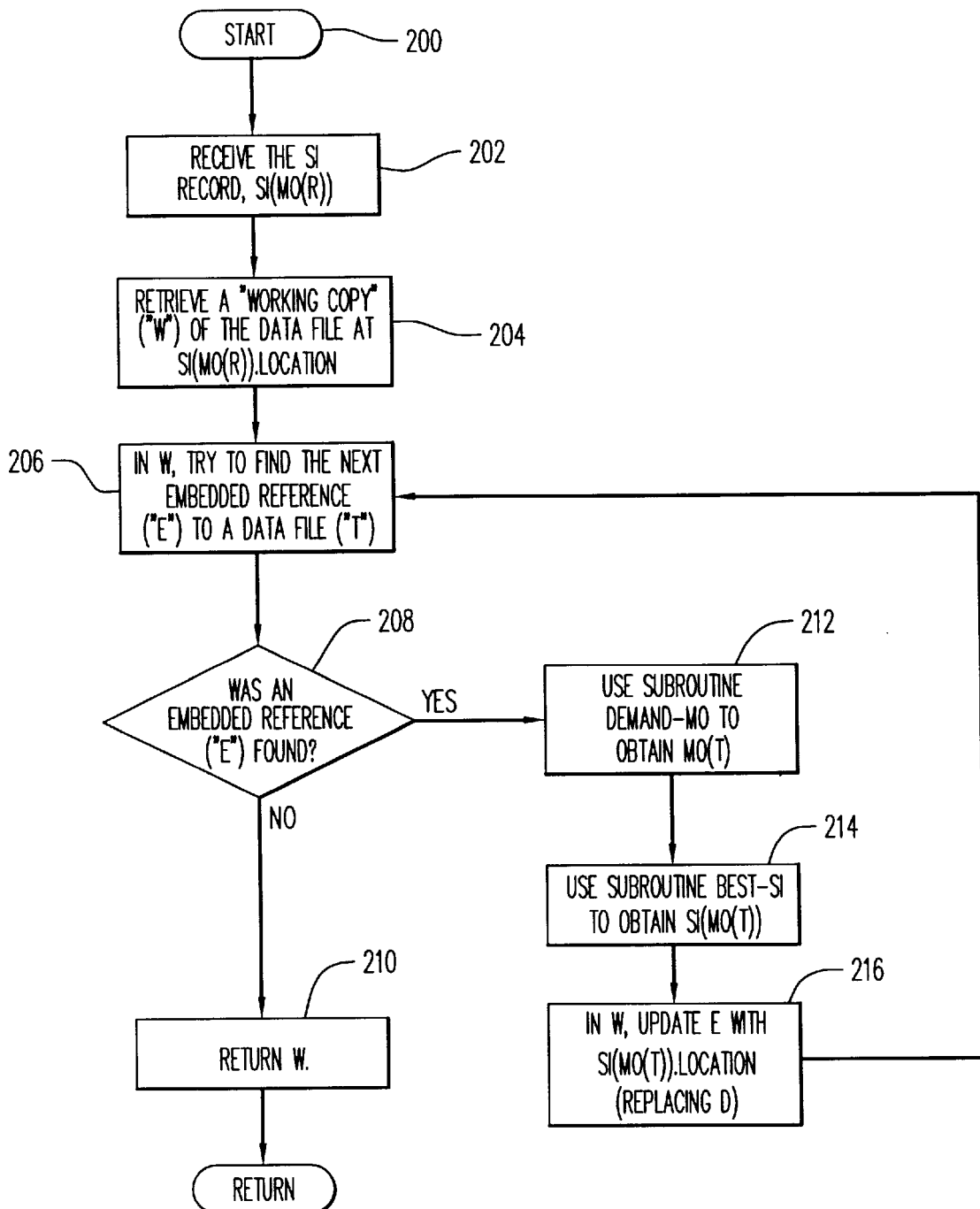


FIG. 6

SUBROUTINE "REWRITE"

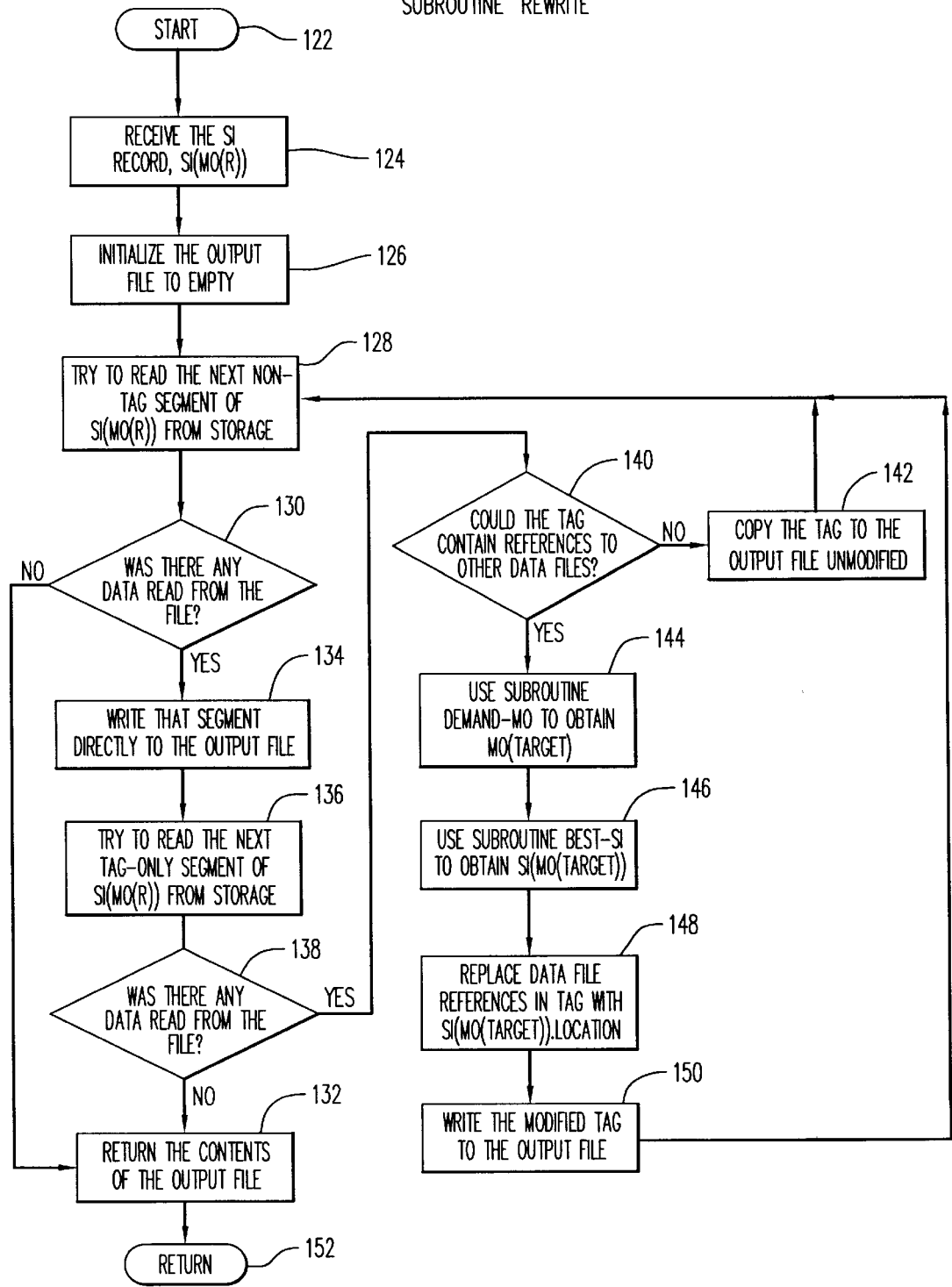
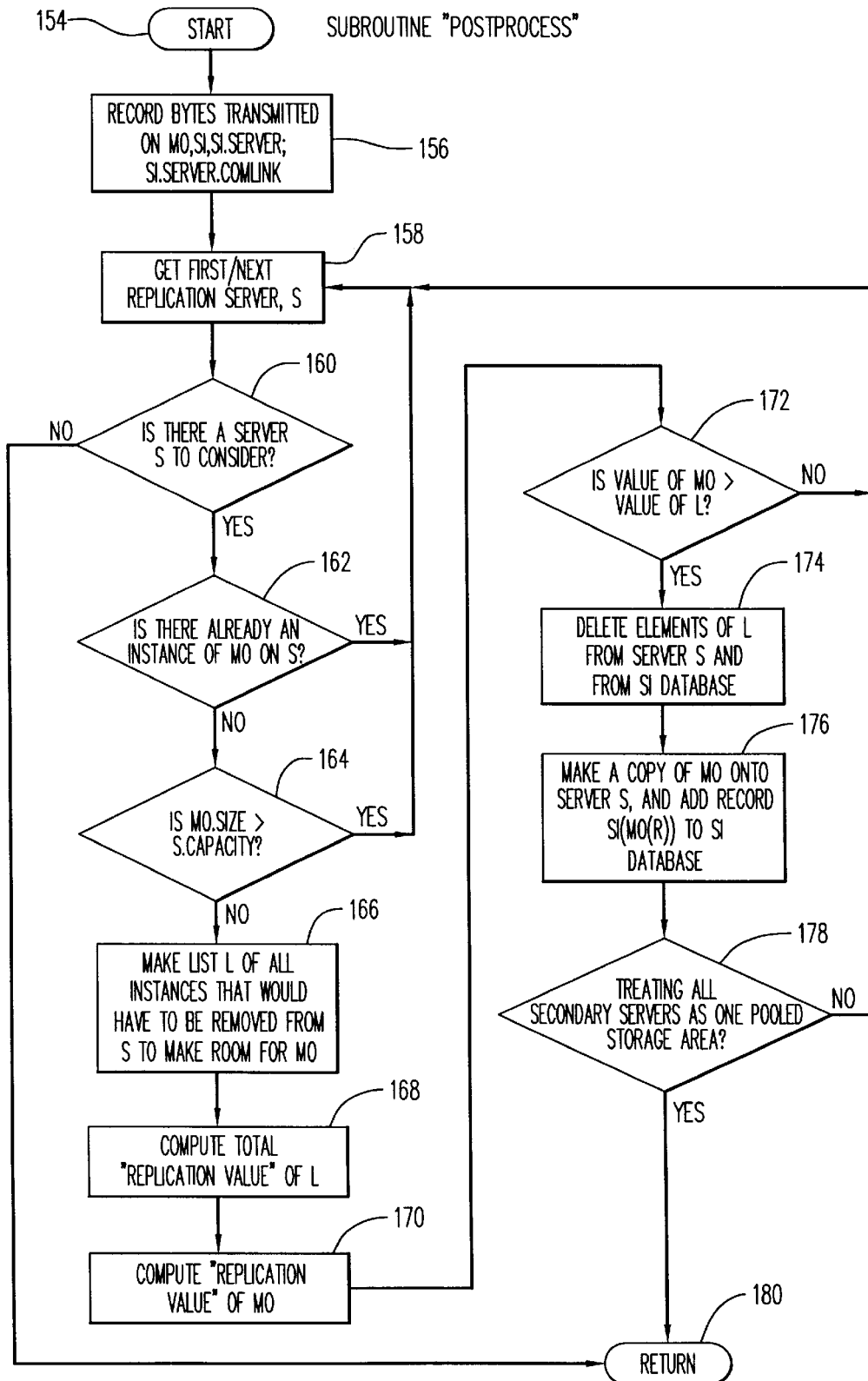


FIG. 7

**WEB SERVING SYSTEM THAT
COORDINATES MULTIPLE SERVERS TO
OPTIMIZE FILE TRANSFERS**

The present application is entitled to the priority of provisional application Ser. No. 60/022,598, filed on Jul. 25, 1996.

The present invention relates generally to a network of computer systems that transfer data files through a network connection, some of which are repeatedly transferred without changes ("static files"), and some of which may be modified for each transfer ("dynamic files"). More particularly, the present invention relates to web servers that are connected to a TCP/IP-capable network and transfer data files of the World Wide Web to computer systems connected to the network. In particular, the present invention is a web serving system that efficiently coordinates the efforts of two or more web servers to transfer data files rapidly through the TCP/IP-capable network while minimizing the cost of such system.

BACKGROUND OF THE INVENTION

A web serving system or web server is a computer system that runs web server software and connects to a communications network, such as the Internet. A publisher may use the web serving system to offer various data files for public access, including text, graphics, audio, video, and other types of data files. Thus, anyone having a computer that is connected to the same network may retrieve these data files offered by the publisher through the use of a standardized communications protocol. One protocol, namely the Hyper Text Transfer Protocol ("HTTP"), is commonly used on TCP/IP-capable networks to transfer data files of the World Wide Web.

Data files of the World Wide Web are transferred from a web server to a web browser computer via a TCP/IP-capable network. The web browser computer typically consists of a computer, web browser software running on the computer, and communications hardware to connect the computer to the network through a communications link. Like wise, the web server typically consists of a computer, web server software running on the computer, storage hardware for storing the web pages, and communications hardware to connect the computer to the network through a communications link. A wide variety of communications links are available for connecting a computer to a TCP/IP-capable network, including an analog telephone connection, 56K line connection, ISDN connection, fractional T1 connection, Full T1 connection, cable-modem connection and satellite connection.

The user may view the contents of a specific data file of a particular web server by operating the web browser software. In particular, when a user wishes to receive the data file, the user operates the web browser computer so that it indicates the network address of the appropriate web server and the name of the data file on the server. The web browser computer then establishes a clear communications channel between the web server and itself, and requests the target data file. Next, the web server retrieves the specified data file from the web server's storage hardware and transmits the contents of the data file through the communications network to the web browser computer. The web browser computer then creates an audio/visual presentation of the data file for the user by activating the various hardware subsystems, such as one or more video monitors and/or audio speakers.

Web browser software has been enhanced to intermingle several pieces of text, graphics, video, audio, and animation together so that browsing or surfing the World Wide Web is an interactive and engaging process. In particular, standardized tags or sequences of text characters are included in the data files to control the placement of intermingled data files on the pages of the World Wide Web. A data file has text data and non-text data in which the standardized tags are included with the text data. The static data includes image data, animation data, video data, computer programs, and other types of non-text data. A data file containing a mixture of text and standardized tags is referred to as web page.

The standardized tags can include additional information, such as the web addresses, data file names of other web servers, and data files of graphic images, video clips, and audio recordings. Thus, most web browser software automatically presents an integrated display of dynamic data and embedded static data when displaying a webpage to the user. Accordingly, two distinct data files are retrieved to create a particular display: the first data file includes the text and tags, and the second data file includes the non-text data.

There are typically over a thousand web browsing users for every web server in operation and, thus, web servers are subjected to numerous requests for data files. Most contemporary web servers are capable of processing this volume of requests, but the web server's communications link quickly becomes saturated. In particular, any given communications link can only transmit a finite number of bits per second, and when a web server using all of the available bandwidth on its communications link to transmit data files, no more data files may be transmitted through that communications link until the transfers in progress are finished. Most communications links cannot sustain more than a dozen concurrent data file transfers, and each transfer can take several seconds.

Because of the high ratio of web browsers to web servers, it is extremely desirable to have a web server that is capable of handling at least ten concurrent data file transfers. Data communications technology, unfortunately, has not evolved as fast as other computer technologies. The least expensive type of communications link is the analog telephone line. However, the extremely constrained capacity of the analog telephone line makes it unacceptable for most web servers. The cost of other higher-speed communications links such as T1 lines increases along with their capacity, and they are far more expensive to setup and maintain than an analog or even ISDN telephone line.

On the other hand, because Full T1 lines are fairly cost effective, Internet Service Providers ("ISPs") have created profitable businesses by purchasing a Full T1 line, operating a web server, and renting out storage space to clients on their web server's storage hardware. These ISPs charge a low monthly rate for renting storage space on their web servers. Any data file that a client places in the rented storage space can be retrieved by web browsers that request them and, thus, clients make their pages and images accessible to a larger number of concurrent browsers without having to pay for the full cost of the full T1 themselves.

However, in order to maintain control and security of their web servers, the ISPs usually limit their clients to publishing static pages. Static pages are only stored and retrieved and, thus, dynamic and interactive pages are not permitted. Dynamic and interactive pages are essentially separate software programs that produce pages as their output. Since the software program is executed each time a web browser computer requests the page, the program may produce a

5,991,809

3

different page each time, specifically for the web browser that made the request. This facility is becoming increasingly popular since it enables the interconnection of web server software and other software programs, such as databases, electronic commerce systems, e-mail systems, stock quote services, etc. Therefore, in order to have dynamic and interactive pages, a publisher cannot rent space from an ISP but must setup and maintain his or her own custom web server which is often prohibitively expensive.

Accordingly, the web serving system of the present invention has the capability of delivering text, graphics, video, audio and interactive multimedia over the Internet's World Wide Web at extremely high speeds, using only inexpensive, readily available hardware, software and services. The system requires no more effort than alternative systems, yet produces the highest performance-to-price ratio of any comparable World Wide Web publishing system.

SUMMARY OF THE INVENTION

The present invention is a collaborative server system capable of providing high speed data transmission of data files across a communications network which, in brief summary, comprises a communications network, a primary server having a primary communications means for connecting the primary server to the communications network, and at least one secondary server having a secondary communications means for connecting the secondary server to the communications network. The primary server and the at least one secondary server include means for storing data files and means for transmitting the data files to the communications network. The data files include static data files and/or dynamic data files. The storage means of the primary server further stores at least one look-up table having specific criteria pertaining to the data files and the primary and at least one secondary servers. The processor means of the primary server is effective to receive a request for specific data files from a network user, to look-up specific criteria in the look-up table pertaining to the specific data files, and to allocate transmission of each specific data file between the primary server and the at least one secondary server based on the specified criteria.

The storage means of the primary server includes static data files and dynamic data files, and the storage means of the secondary server includes a duplicate of the static data files. Thus, the processor means includes means for transferring the duplicate of the static data files from the storage means of the primary server to the storage means of the secondary server. Accordingly, the processor means of the primary server is capable of providing the dynamic files of the primary server and the duplicate of the static data files of either the primary server or the secondary server to the communications network in response to a single request for the static data files and dynamic data files of the primary server.

In addition, it is preferred that the secondary communications means of the secondary server transmits and receives communications signals to and from certain user's computers at faster rates, with lower latency, more cost-effectively, or more efficiently than that of the primary communications means of the primary server. In particular, the primary communications means of the primary server is a lower-cost connection to the communications network (i.e., an analog communication line), and the secondary communication means is a connection with better bandwidth, latency, cost-effectiveness, and efficiency (i.e., a digital T1 communication line).

4

Further, the system includes at least one computer system that is connected to the communications network and has the capability of sending a request to the primary server. In addition, the processor means of the primary server includes means for determining an optimum server from the group of servers including the primary and secondary server to transmit the duplicate of the static data files to the computer system when two or more servers are available. The criteria for determining which one of the servers shall transmit includes transmission speed and available capacity of the primary communications means, proximity of the computer system to each server, availability of each server, version of the duplicate of the data file on each server and financial cost of transmitting data from each server.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and still further objects and advantages of the present invention will be more apparent from the following detailed explanation of the preferred embodiments of the invention in connection with the accompanying drawings:

FIG. 1 is a diagrammatic view of the preferred web serving system in accordance with the present invention;

FIG. 2 is a flow diagram representing the operation of the primary web server's computer when used with the supplemental web server software of FIG. 1;

FIG. 3 is a flow diagram of the DEMAND MO subroutine of FIG. 2;

FIG. 4 is a flow diagram of the BEST SI subroutine of FIG. 2;

FIG. 5 is a flow diagram of the REWRITE subroutine of FIG. 2;

FIG. 6 is a preferred flow diagram of the REWRITE subroutine of FIG. 2; and

FIG. 7 is a flow diagram of the POSTPROCESS subroutine of FIG. 2.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to the drawings, and in particular, to FIG. 1, there is provided a web serving system of the preferred embodiment which is generally represented by reference numeral 10. Web serving system 10 has the capability of delivering text, graphics, audio recordings, video recordings, computer animation, and computer programs and data to remote computer systems and storage devices through a distributed data network at extremely high speeds, often in excess of 300,000 bits per second, in a way that is interoperable with the World Wide Web's hypertext protocol. In particular, web serving system 10 combines a low-cost communications channel of a primary server with a more desirable communications channel of one or more secondary servers as well as computer storage and retrieval services for both the low speed and high speed channels. By coordinating data retrieval from the various servers, the primary server is able to control the data files sent by each server so that the speed in which data files are delivered to a target browser computer is maximized while retaining the ability to provide dynamic and interactive pages (i.e., dynamic data files). In addition, the setup and maintenance costs of web serving system 10 are substantially less than that of conventional web serving systems since its secondary web server takes advantage of the low cost of renting or otherwise using storage space in existing servers and its primary web server takes advantage of lower-cost, lower-

bandwidth communications links including analog telephone lines and lower-speed digital connections.

When used with an analog telephone line, preferred web serving system 10 shown in FIG. 1 is capable of fulfilling requests for over 100,000 data files per day at a minimal cost. The available capacity of the system is dependent on a number of factors and conditions, but the system can reliably sustain performance in the 400 Kbps range and process many transfers concurrently. Also, the monthly cost per concurrent connection is much lower than that of a conventional web serving system that uses a 56K line connection, ISDN connection, fractional T1 connection, or Full T1 connection.

As shown in FIG. 1, preferred web serving system 10 includes one or more web browser computer systems 12, a primary web server 14 and one or more local web browser computer system 15 connected on a local area network (LAN) 35, one or more secondary web servers 16 and one or more satellite web browser computer system 15 connected on a LAN 51, and a communications network 18. Web browser computer system 12 may be a web user connected on a local area network (LAN) 35 with primary web server 14 (local user), connected on a LAN 51 with secondary web server 16 (satellite user) or not connected to either LAN 35 or LAN 51 (remote user).

Each web browser computer system 12 includes a user's computer 20, web browser software 22 residing in computer 20, communications hardware 24 connected to computer 20, and communications link 26 connecting communications hardware 24 to communications network 18. In particular, user's computer 20 may be any type of computer system that is capable of running user's web browser software 22 and connecting to communications network 18. For example, user's computer 20 may be a personal computer or workstation that includes a memory portion for storing user's web browser software 22 and a processor for executing web browser software 22 so that the user may interact with software 22. Also, user's computer 20 is connected to user's communications hardware 24 which, in turn, is linked to communications network 18 by user's communications link 26. For example, a modem may be used to link user's computer 20 to an Internet Service Provider ("ISP") through a telephone line. Since a wide variety of different computer systems may connect to the network, it is to be understood that user's computer 20 of the present invention is not limited to the personal computer or workstation examples described above. Note that web browser computer systems 12 which are connected to either LAN 35 or LAN 51 respectively employ the same communications hardware and link as primary server system 14 and secondary server system 16.

Primary web server 14 includes a primary computer 28, storage hardware 30 connected to computer 28, web server software 32 and supplemental web server software 34 residing in computer 28, communications hardware 36 connected to computer 28, and communications link 38 providing a connection between communications hardware 36 and communications network 18. Also, primary storage hardware 30 may reside internal or external to primary computer 28 and is capable of storing primary data files, such as original dynamic data files 40 and original static data files 42. Dynamic data files 40 include standardized tags as well as text data, whereas static data files 42 typically include non-text data, such as image data, animation data, video data, audio data, and computer programs. Primary data files (40, 42) are commonly transmitted over packet-switched communications network 18 using TCP/IP protocols and

read by user's web browser software 22. Primary computer 28 may be any type of computer system that is capable of running primary web server software 32 and supplemental web server software 34, and connecting to communications network 18. A wide variety of computer systems, ranging from mainframe computer to personal computers, are capable of running web server software 32 with supplemental web server software 34 and, thus, primary computer 28 is affordable and readily available. For example, primary computer 28 may be a personal computer or workstation that includes a memory portion for storing primary web server software 32 and supplemental web server software 34, and a processor for executing both software. Also, similar to web browser computer system 12, primary computer 28 is connected to primary communications hardware 36 which, in turn, is linked to communications network 18 by primary communications link 38.

In addition, the present invention includes one or more secondary web servers 16. Although a secondary web server 16 may be purchased and setup for the present invention, the services of such secondary web server 16 is readily available to avoid their high setup and maintenance costs, which is a feature of the present invention. For example, one may rent space on a secondary web server from an Internet Service Provider for a low fee, or arrange to use space on a secondary web server owned by an organization or university.

Typically, each secondary web server 16 includes a secondary computer 44, storage hardware 46 connected external or provided internal to secondary computer 44, web server software 48 and remote storage access software 50 residing in secondary computer 44, communications hardware 52 connected to secondary computer 44, and communications link 54 connecting secondary computer 44 to communications network 18. Secondary storage hardware 46 is capable of storing data files 56 that are duplicates of the original data files (40, 42) in primary storage hardware 30 of primary web server 14. Also, secondary communications link 54 has superior operating characteristics than that of primary communications link 38. For example, as shown in FIG. 1, secondary communications link 54 is a Full T1 line that has higher performance capacity than primary communications link 38 which is an analog telephone line. Also, similar to primary computer 28, secondary computer 44 may be any type of computer system, ranging from personal computer to mainframe computers, that is capable of running secondary web server software 48 and remote storage access software 50, and connecting to communications network 18. Further secondary computer 44 is connected to secondary communications hardware 52 which, in turn, is linked to communications network 18 by secondary communications link 54.

Computer 28 of the primary web server 14 executes supplemental web server software 34 to perform the critical function of coordinating file transfers by primary web server 14 and secondary web server or servers 16. Computer 28 and supplemental web server software 34 exploit the fact that the ultimate audience, namely the users who accessing primary web server 14, for data files (40, 42) is not involved in the retrieval of each data file individually. In actuality, the task of retrieving data files, particularly static data files 42 that are embedded in pages, is performed automatically by the user's web browser software, such as web browser software 22.

Primary computer 28, when used with supplemental web server software 34, adds significant and powerful capabilities to primary web server software 32. In particular, primary

computer 28 identifies the static files (i.e., typically non-text data files) that are consuming most of the capacity of primary communications link 38, copies those capacity-consuming static data files 42 to one or more secondary web servers 16 (which have higher capacity communications links), and then dynamically re-writes requested pages so that the embedded static data files are retrieved directly from the best or optimal web server each time. Primary computer 28 also records every request for a data file (40, 42), builds a statistical model of how the capacity of primary communications link 38 is being consumed by the transmission of the various data files (40, 42), determines which static data files 42 are consuming the most capacity, uses standardized communication protocols to communicate through the channel between primary and secondary web servers (14, 16), uses the communication channel to copy static data files 42 from primary storage hardware 30 to secondary storage hardware 46, and tracks the progress and location of all duplicated files 56 through the use of a database.

In addition, primary computer 28 responds to requests for pages stored on the primary storage hardware 30 by reading the data file (40, 42) from the primary storage hardware 30, finding all tags in the data file (40, 42) that refer to static data files 42 stored on primary web server 14, examining the database to retrieve a list of secondary web servers 16 and data files 56 where instantiation of static data file 56 referred to by each tag may be found. Primary computer 28 also responds to requests for pages stored on primary storage hardware 30 by determining, for each static data file 42 referred to by the page's tags, which available instantiation would be the best instantiation to transmit to web browser computer system 12. Considerations for determining the best instantiation includes the transmission speed of each web server's communications link (38, 54), the currently available capacity of each web server's communications link (38, 54), the proximity of web browser computer system 12 to each web server (14, 16) given the topology of the communications network 18, the current availability or operability of each secondary web server 16, the version of the data file 42 available on each web server (14, 16) in which only instantiations that are current are eligible, the cost of transmitting the data file from each server, the response latency of each server with respect to the requesting computer and the efficiency of transmitting the data file from each server. Primary computer 28 further responds to requests for pages stored on the primary storage hardware 30 by creating a re-written page, in which each tag that refers to a particular data file 42 is replaced with a tag that is functionally identical except that the tag now refers to the best instantiation rather than to original data file 42. Also, the re-written page is transmitted through the primary connections link to web browser computer system 12, statistical information about the transmission of the page and the supposed transmission of any static data file 42 whose best instantiation is not on primary web server 14 is recorded, various statistics are used to update the database with revised information about the usage levels and patterns of data files (40, 42, 56) on both primary web server 14 and secondary web servers 16, and statistical information in the database is used to guide primary computer 28 in optimizing the distribution of data file instantiations across multiple servers.

Once static data files 42 have been copied to one or more secondary web servers 16, only original dynamic data files 40, particularly their text-and-tags information, need to be transferred through primary communications link 38. Most static data files 56 that are requested from outside the local

area network are transmitted over the higher-speed, higher-capacity, lower-latency or more efficient links of secondary communications hardware 52. If one secondary web server 16, whose communications link 54 is a Full T1 line, is used, the speed at which the primary computer 28 can deliver pages and static data files can be as high as about 1200 Kbps for short bursts and tends to run consistently about 300 Kbps to about 400 Kbps. Thus, by combining the resources of a low-cost, low-bandwidth communications link 38 and some low-cost storage space on one or more secondary web servers 16 with high speed, high capacity communications links 54, the result is a web serving system which is fast, interactive and inexpensive.

It is important to note that the present invention provides high capacity performance, reliably sustaining about 400 Kilobits per second of capacity, while providing the capability of handling dynamic and interactive pages and benefiting from the economical startup and maintenance costs of a low cost, low bandwidth communications link. In particular, when used with an analog telephone line, the present invention is capable of delivering performance that is comparable to a system having a fractional T1 connection and surpasses a system having 56K line or ISDN connection. Therefore, for the present invention, an expensive digital communications link, such as fractional T1 connection or Full T1 connection, is not required for high capacity performance. In fact, the primary communications link may be an analog telephone line connected to the primary communications hardware, such as a modem, and yet provide performance that is comparable to a fractional T1 connection and outperform a 56K line or ISDN connection.

Referring to FIG. 2, there is provided the main flow chart, beginning with step 60, for the process performed by primary computer 28, when using supplemental web server software 34, for coordinating the file transfer functions of primary web server 14 and secondary web server or servers 16. When a new request for a data file (40, 42) is received by web server software 32 as shown in step 62, primary computer 28 starts by examining the name of the requested file as using subroutine DEMAND MO shown in step 64. The coordinating means then selects the instance with the highest quality that is associated with this master object by using subroutine BEST SI as shown in step 66.

If the best instance is not associated with the primary server, then a redirect response indicating the location of the instance master object is prepared as shown in steps 68 and 70. If the best instance is associated with the primary server but the requested data file is not a page that may contain tags or reference to other data files, a direct response including the contents of the instance master object is prepared as shown in steps 72 and 74. If the best instance is associated with the primary server and the requested data file contains tags or reference to other data files, then a revised version of the contents of the instance master object is generated using subroutine REWRITE as shown in step 76. Then, as shown in step 78, a direct response including the rewritten contents of the instance master object is prepared. Next, in all cases, the contents of the instance found on the primary server are transmitted to the web browser that requested it as shown in step 80. Finally, statistical information is recorded using subroutine POSTPROCESS and primary computer 28 stops execution as shown in steps 82 and 84.

Referring to FIG. 3, the subroutine DEMAND-MO (step 64 of FIG. 2) begins at step 86 and responds to the request for the data file (40, 42) received by web server software 32 as shown in step 88. Primary computer 28 then searches the database for a master object record that matches the name

given as shown in step 90. If there is a matching master object record in the database, then the coordinating means returns the master object record as shown in steps 92 and 94. If a matching record is not in the database, a new master object record with the name of the given file is created in the database as shown in step 96. Next, a new statistics record is created for this master object, and an instance record with the name of the given file is created in the instance database as shown in step 98. Also, the instance record is marked as primary and local and the instance is associated with primary web server 14. Finally, as shown in step 94, the new master object record is returned and, as shown in step 100, subroutine DEMAND MO returns to the main process of FIG. 2.

Referring to FIG. 4, there is provided the BEST SI subroutine shown as step 66 in FIG. 2, starting with step 102. Primary computer 28 initially receives a given master object and retrieves the list of instances associated with the given master object from the database as shown in steps 104 and 106. As shown in steps 108, primary computer 28 determines whether there are any more matching records. If no more matching records exist then the primary computer 28 returns the instance master object having the lowest serving time and/or highest quality score, and the BEST SI subroutine returns to the main process of FIG. 2 as shown in steps 110 and 120.

If there are any more matching records, such record is retrieved as shown in step 112. Then, the serving time is calculated by considering the transmission time of the data file from the given instance, given the current load on the server for the instance and the load and maximum speed of the communications link (38, 54) for that server (14, 16). The quality score is set to be an inverse function of the time required, and scaled by the topological distance between the web browser computer system 12 making the request and the server (14, 16) for this instance. Cost, latency and efficiency are also considered part of the quality score. If the serving time is faster than any other serving time considered so far for this master object, then the instance is designated as having the lowest serving time, and primary computer 28 searches for the next record matching the master object in the database as shown in steps 116 and 118. Otherwise, if the serving time is not the fastest one considered, then primary computer 28 simply goes back to searching for the next record matching the master object in the database as shown in steps 116.

Referring to FIG. 5, there is provided the REWRITE subroutine shown as step 76 in FIG. 2, starting with step 200. Initially, the instance record is received, and a working copy of the data file of the instance record location is retrieved, as shown in steps 202 and 204. Primary computer 28 then analyzes the working copy of the data file for the next embedded reference to a data file as shown in step 206. Primary computer 28 determines whether an embedded reference was found in step 208. If not, the working copy is returned without any modifications as shown in step 210. If an embedded reference is discovered, the DEMAND MO subroutine is used to find the master object record for the embedded data file as shown in step 212. Next, the BEST SI subroutine is used to find the best instance for that master object as shown in step 214. Then, in the working copy of the data file, the reference to the original Web Server (i.e., an address code) and Data File names are replaced with the names of the web server and data file where the best instance is located as shown in step 216. If there is more in the data file to read, then primary computer 28 will try to read the next embedded reference to a data file. Finally, the primary

computer 28 transmits the contents of working file to the web browser that requested it as shown in step 210.

Referring to FIG. 6, there is provided a preferred embodiment of the REWRITE subroutine shown as step 76 in FIG. 2, starting with step 122. Initially, the instance record is received, and the output file is initialized so that it is empty as shown in steps 124 and 126. The next segment of text is then read from the data file as shown in step 128. Then in step 130, primary computer 28 determines whether any data was read from the file. If not, the contents of the output file are returned as shown in step 132. However, if data was read from the file, then all non-tag text is copied to the page output buffer as shown in step 134. When a tag is encountered, primary computer 28 tries to read the next tag-only segment of the incident master object from storage as shown in step 136. Next, as shown in step 138, the primary computer 28 determines whether data was read from the file. If not, then the contents of the output file are returned as shown in step 132. As shown in steps 138 and 140, if data was read from the file, the primary computer 28 determines whether there is a reference to another data file. If none is found, the tag is copied to the output file without modification as shown in step 142. However, if the tag could contain such references, the DEMAND MO subroutine is used to find the master object record for the name found as shown in step 144. Next, the BEST SI subroutine is used to find the best instance for that master object as shown in step 146. Then, in the tag, the reference to the original Web Server and Data File names are replaced with the names of the web server and data file where the best instance is located as shown in step 148. The tag is then appended to the page output buffer as shown in step 150. If there is more in the data file to read, then primary computer 28 will try to read the next non-tag segment. Finally, the primary computer 28 transmits the contents of the page output buffer to the web browser that requested it as shown in step 132.

Referring to FIG. 7, there is provided the POSTPROCESS subroutine shown as step 82 in FIG. 2, starting with step 154. Primary computer 28 updates the statistical database records for the instance, the master object, the server, and the communications links appropriate to the instance that was selected as shown in step 156. Then, the first replication server is obtained as shown in step 158, and a determination is made as to whether there is a server to consider as shown in step 160. If not, then subroutine POSTPROCESS returns to the main process of FIG. 1. If a server to consider does exist, then a determination is made of whether there is already an instance of this master object on this server as shown in step 162. If so, the next replication server is considered as shown in step 158. If there is not an instance of the master object on the server, primary computer 28 determines whether the master object is larger than the total storage capacity of this server as shown in step 164. If so, the next replication server is considered as shown in step 158.

If the master object is not larger than the total storage capacity of the server, a list is made of the instances, if any, that would have to be deleted from this server in order to accommodate the master object under consideration as shown in step 166. The total replication value of the list and the replication value of the master object are computed as shown in steps 168 and 170. The total replication value of the list is compared to the replication value of the master object as shown in step 172. If the replication value of the given master object is not greater than the total replication value of the list, then the next replication server is examined as shown in step 158. If the replication value of the given

master object is greater than the total replication value of the list, then the remote instances on the list are deleted. Specifically, elements of the list are deleted from the given server and from the instance database as shown in step 174, and a copy of the master object is made onto the given server and the record of instance master object is added to the instance database as shown in step 176. If all secondary web servers 16 are to be treated as one pooled storage area, then the POSTPROCESS subroutine returns to the main process of FIG. 2 as shown in step 178. If not, then more servers are examined as shown in step 158.

The present invention having been described with particular reference to the preferred forms thereof, it will be obvious that various changes and modifications may be made therein without departing from the spirit and scope of the invention as defined in the appended claims.

What is claimed is:

1. An apparatus that is a member of a group of predetermined devices connected to a network, for receiving a request for data files from a network user and allocating transmission of the data files between said predetermined devices, said apparatus comprising:

memory means for storing a data file and a look-up table having criteria pertaining to said data file and said predetermined devices; and

data allocation means which is capable of:

- (i) receiving a request for said data file from a network user,
- (ii) looking up said criteria in said look-up table pertaining to said data file, and
- (iii) allocating transmission of said data file between said predetermined devices based on said criteria; and

means for transmitting data to said network user, wherein said data file is selected from a group consisting of a dynamic data file and a static data file.

2. The apparatus as recited in claim 1, wherein said data file is a dynamic data file, and wherein said data allocation means causes said transmitting means to transmit said data file to said network user.

3. The apparatus as recited in claim 1, wherein said data file is a static data file, and wherein said data allocation means selects an optimum device from said predetermined devices and causes said network user to request said data file from said optimum device.

4. The apparatus as recited in claim 3, wherein said data allocation means transmits an address code of said optimum device to said network user.

5. The apparatus as recited in claim 4, wherein said network user, responsive to receipt of said address code, requests said data file from said optimum device.

6. The apparatus as recited in claim 1, further comprising means for transferring a copy of said data file stored in said memory means to any of said predetermined devices.

7. The apparatus as recited in claim 1, further comprising means for updating said criteria in said look-up table of said memory means.

8. The apparatus as recited in claim 1, wherein said criteria includes a latency of a connection between each of said predetermined devices and said network user.

9. The apparatus as recited in claim 1, wherein said criteria includes an estimated cost of transmitting said data file from each of said predetermined devices to said network user.

10. The apparatus as recited in claim 1, wherein said criteria includes a transmission bandwidth of each of said predetermined devices.

11. The apparatus as recited in claim 1, wherein said criteria includes a current capacity of each of said predetermined devices to transmit data files to said network user.

12. The apparatus as recited in claim 1, wherein said criteria includes a version type of data files retrievable by each of said predetermined devices.

13. The apparatus as recited in claim 1, wherein said data file includes an embedded reference to an additional data file.

14. The apparatus as recited in claim 13, wherein said data allocation means is further capable of looking up criteria in said look-up table pertaining to said additional data file, and allocating transmission of said additional data file between said predetermined devices, based on said criteria pertaining to said additional data file.

15. The server system as recited in claim 13, wherein said embedded reference includes an address code for said additional data file, said allocation means having means for rewriting said address code.

16. A collaborative server system for providing high speed data transmission of data files across a network, comprising:

a primary server and a secondary server, both connected to a network across a communication medium, said primary server comprising:

(a) memory means for storing a data file and a look-up table having specific criteria pertaining to said data file and said primary server and said secondary server;

(b) data allocation means which is capable of:

- (i) receiving a request for said data file from a network user,
- (ii) looking up said criteria in said look-up table pertaining to said data file, and
- (iii) allocating transmission of said data file between said primary server and said secondary server based on said criteria; and

(c) means for transmitting data across said network, said secondary server comprising: secondary memory means for storing said data file, and means for transmitting data across said network,

wherein said data file is selected from a group consisting of a dynamic data file and a static data file.

17. The server system as recited in claim 16, wherein said data file is a dynamic data file, and wherein said data allocation means causes said transmitting means of said primary server to transmit said data file to said network user.

18. The server system as recited in claim 16, wherein said data file is a static data file, and wherein said data allocation means selects an optimum server from a group including said primary server and said secondary server and causes said network user to request said data file from said optimum device.

19. The server system as recited in claim 18, wherein said data allocation means transmits an address code of said optimum device to said network user.

20. The server system as recited in claim 19, wherein said network user, responsive to receipt of said address code, requests said data file from said optimum device.

21. The server system as recited in claim 16, wherein said primary server further comprises means for transferring a copy of said data file stored in said primary memory means to said secondary memory means of said secondary server.

22. The server system recited in claim 16, wherein said primary server further comprises means for updating said criteria in said look-up table of said primary memory means.

23. The server system as recited in claim 16, wherein said criteria includes a latency of a connection between said primary server and said secondary server and said network user.

5,991,809

13

24. The server system as recited in claim 16, wherein said criteria includes an estimated cost of transmitting said data file from said primary server and said secondary server to said network user.

25. The server system as recited in claim 16, wherein said criteria includes a transmission bandwidth of said primary server and said secondary server.

26. The server system as recited in claim 16, wherein said criteria includes a current capacity of said primary server and said secondary server to transmit data files to said network user.

27. The server system as recited in claim 16, wherein said criteria includes a version type of data files retrievable by said primary server and said secondary server.

28. The server system as recited in claim 16, wherein said data file includes an embedded reference to an additional data file.

29. The server system as recited in claim 28, wherein said data allocation means is further capable of: looking up criteria in said look-up table pertaining to said additional data file, and allocating transmission of said additional data file between said primary server and said secondary server, based on said criteria pertaining to said additional data file.

30. The server system as recited in claim 28, wherein said embedded reference includes an address code for said additional data file, said data allocation means having means for rewriting said address code.

31. The server system as recited in claim 16, wherein said primary server is connected to said network across a slower transmission medium than said secondary server.

32. The server system as recited in claim 31, wherein said primary server is connected to said network across an analog transmission line and said secondary server is connected to said network across a faster digital transmission line.

33. The server system as recited in claim 16, wherein said network user includes means for requesting and receiving said data file, via said network.

34. The server system as recited in claim 33, wherein said network user further includes means for generating a web page from said data file.

14

35. The server system as recited in claim 16, wherein said network user and said primary server are connected in a local area network (LAN), said LAN being connected to said network.

36. The server system as recited in claim 16, wherein said network user and said secondary server are connected to a local area network (LAN), said LAN being connected to said network.

37. An apparatus that is a member of a group of predetermined devices connected to a network, for receiving a request for data files having embedded therein references to additional data files from a network user and allocating transmission of all of the data files between said predetermined devices, said apparatus comprising:

memory means for storing a data file and an additional data file and a look-up table having criteria pertaining to said data file and said additional data file and said predetermined devices, said data file having an embedded reference to said additional data file; and

data allocation means which is capable of:

- (i) receiving a request for said data file from a network user,
- (ii) looking up criteria in said look-up table pertaining to said data file and said additional data file, and
- (iii) allocating transmission of said data file and said additional data file between said predetermined devices based on said criteria; and

means for transmitting data to said network user,

wherein said data file and said additional data file are selected from a group consisting of a dynamic data file and a static data file.

38. The apparatus as recited in claim 37, wherein said embedded reference includes an address code for said additional data file, said data allocation means having means for rewriting said address code.

* * * * *

EXHIBIT I

1

CONFIDENTIAL * ATTORNEYS EYES ONLY

IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

GIRAFa.COM, INC.,

Case No. 07-787-(SLR)

Plaintiff,

v.

CONFIDENTIAL
COPY

AMAZON WEB SERVICES LLC,

AMAZON.COM, INC., ALEXA

INTERNET, INC., IAC SEARCH &

MEDIA, INC., SNAP TECHNOLOGIES,

INC., YAHOO!, INC., EXALEAD S.A.,

and EXALEAD, INC.,

Defendants.

VIDEOTAPED DEPOSITION OF BRAD A. MYERS

Volume I

Washington, DC

Friday, April 18, 2008

8:00 a.m.

Job No. 1-126469

Pages 1-361

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME 1
CONDUCTED ON FRIDAY, APRIL 18, 2008

2

CONFIDENTIAL * ATTORNEYS EYES ONLY

Reported by: Linda S. Kinkade, CSR, RMR, CRR

Videographer: Scott Forman, L.A.D. Reporting

Videotaped Deposition of BRAD A. MYERS, held
at the offices of:

Sughrue Mion, PLLC

2100 Pennsylvania Avenue, Northwest

Washington, D.C. 20037-3213

Pursuant to applicable Rules of Civil
Procedure, before Linda S. Kinkade, CSR, RMR, CRR, a
Notary Public, in and for the District of Columbia.

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME I
CONDUCTED ON FRIDAY, APRIL 18, 2008

3

CONFIDENTIAL * ATTORNEYS EYES ONLY

APPEARANCES:

On Behalf of Plaintiff:

JOHN F. RABENA, ESQUIRE

WILLIAM H. MANDIR, ESQUIRE

CHANDRAN IYER, ESQUIRE

TREVOR HILL, ESQUIRE

Sughrue Mion, PLLC

2100 Pennsylvania Avenue, Northwest

Washington, D.C. 20037-3213

Telephone: (202) 663-7472

On Behalf of Defendant AMAZON WEB SERVICES, LLC,
AMAZON.COM, INC., ALEXA INTERNET, INC.:

THOMAS G. PASTERNAK, ESQUIRE

R. DAVID DONOGHUE, ESQUIRE

DLA Piper

203 North LaSalle Street

Suite 1900

Chicago, Illinois 60601

Telephone: (312) 368-4000

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME I
CONDUCTED ON FRIDAY, APRIL 18, 2008

4

CONFIDENTIAL * ATTORNEYS EYES ONLY

APPEARANCES (continued):

On Behalf of Defendant SNAP TECHNOLOGIES:

MARK D. NIELSEN, Ph.D., ESQUIRE

Attorney at Law

Cislo & Thomas, LLP

1333 2nd Street, Suite 500

Santa Monica, California 90401

Telephone: (310) 451-0647

On Behalf of Defendant IAC SEARCH & MEDIA, INC.:

ALISON MONAHAN, ESQUIRE

Quinn Emanuel Urquhart Oliver & Hedges, LLP

50 California Street

22nd Floor

San Francisco, California 94111

Telephone: (415) 875-6394

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME I
CONDUCTED ON FRIDAY, APRIL 18, 2008

5

CONFIDENTIAL * ATTORNEYS EYES ONLY

APPEARANCES (continued):

On Behalf of Defendants EXALEAD S.A. and EXALEAD,
INC.:

HAROLD V. JOHNSON, ESQUIRE

Brinks Hofer Gilson & Lioné

NBC Tower, Suite 3600

455 N. Cityfront Plaza Drive

Chicago, Illinois 60611

Telephone: (312) 321-4200

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME 1
CONDUCTED ON FRIDAY, APRIL 18, 2008

285

1 CONFIDENTIAL * ATTORNEYS EYES ONLY

2 been informed about certain aspects of the law that 16:07:05
3 are relevant to your analysis and opinion. Do you see 16:07:08
4 that? 16:07:11

5 A Yes. 16:07:11

6 Q Did you do anything to confirm that that 16:07:12
7 information provided to you was accurate? 16:07:14

8 A No. 16:07:16

9 Q Paragraph 38. 16:07:17

10 A Okay. 16:07:42

11 Q There is just a phrase in there I want to 16:07:48
12 talk about briefly, annotated web page. What's your 16:07:50
13 understanding of what that term means in the context 16:07:53
14 of Girafa's patent? 16:07:55

15 A Well, it explains it in the next clause, 16:08:00
16 which preferably includes the web page having 16:08:03
17 alongside it images of home pages linked with the web 16:08:06
18 page. 16:08:11

19 Q So the annotated web page would show the 16:08:13
20 thumbnails of the hyperlinked home pages on it; is 16:08:16
21 that accurate? 16:08:20

22 A Not necessarily. I think it's pretty clear 16:08:20

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME I
CONDUCTED ON FRIDAY, APRIL 18, 2008

286

1 CONFIDENTIAL * ATTORNEYS EYES ONLY

2 that the first embodiment, the annotations are 16:08:27
3 external, I mean -- yeah, the annotations are external 16:08:32
4 to the web page itself, that they are alongside it, 16:08:37
5 like it says here. 16:08:40

6 Q Okay. I probably didn't phrase that 16:08:41
7 question properly. Well, it uses the word "annotated 16:08:43
8 web page," though, meaning -- doesn't that mean it's 16:08:49
9 one web page and it's got the thumbnails alongside 16:08:53
10 whatever it is, the search results or other 16:08:57
11 hyperlinks? 16:08:59

12 A Well, let's look. Let's look at the 16:09:03
13 context where it's talking about that. 16:09:06

14 Q Are you talking about column 5, lines 61 to 16:09:19
15 65? 16:09:22

16 A Sure. That's where this is quoted from. 16:09:23
17 So since it's specifically referencing FIG. 1, and an 16:09:26
18 annotated web page 110 which includes the web page 101 16:09:39
19 having alongside of it images 112 of home pages linked 16:09:52
20 with web page 101. 16:09:57

21 So, I think this part of the specification is 16:09:58
22 being a little loose with the word "web page" because 16:10:01

287

1 CONFIDENTIAL * ATTORNEYS EYES ONLY

2 it's saying both 110 and 101 are web pages; whereas, 16:10:09
3 it looks like 110 is pointing to the entire browser 16:10:14
4 where you can see that the column 112 of images is 16:10:21
5 alongside the original web page 101, like it says. 16:10:25

6 So, I think, when it says an annotated web page 16:10:30
7 110, it's really referring to this entire image, which 16:10:34
8 is labeled 110 in the figure. 16:10:41

9 Q So it's the original web page and then 16:10:43
10 there is this separate side panel, I guess, with the 16:10:46
11 thumbnails of the various hyperlinks referenced in the 16:10:48
12 original web page, is that accurate, as to what an 16:10:51
13 annotated web page is here? 16:10:54

14 A In this particular phrase referencing this 16:10:55
15 particular figure, I think that's what it refers to. 16:11:00

16 Q Okay. And if you turn the page of your 16:11:02
17 report to, I guess it's going to page 8 of your 16:11:17
18 report, that picture at the top of page 8 of your 16:11:20
19 report is the annotated web page, is that correct, 16:11:23
20 that's shown in FIG. 1 of the patent? 16:11:28

21 A Yes, as it was just described. 16:11:29

22 Q Okay. Now, in paragraph 39 you're talking 16:11:31

CONFIDENTIAL VIDEOTAPED DEPOSITION OF BRAD ALLAN MYERS, PH.D. - VOLUME I
CONDUCTED ON FRIDAY, APRIL 18, 2008

359

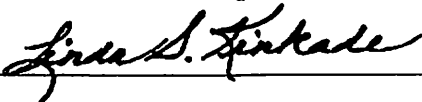
CONFIDENTIAL * ATTORNEYS EYES ONLY

CERTIFICATE OF SHORTHAND REPORTER - NOTARY PUBLIC

I, LINDA S. KINKADE, Registered Professional
and Registered Merit Reporter, Certified Shorthand
Reporter and Certified Realtime Reporter, the officer
before whom the foregoing proceedings were taken, do
hereby certify that the foregoing transcript is a true
and correct record of the proceedings; that said
proceedings were taken by me stenographically and
thereafter reduced to typewriting; and that I am
neither counsel for or related to, nor employed by any
of the parties to this case and have no interest,
financial or otherwise, in its outcome.

IN WITNESS WHEREOF, I have hereunto set my hand
and affixed my notarial seal this 20th day of April,
2008.

My commission expires: July 15, 2012



NOTARY PUBLIC IN AND FOR

THE DISTRICT OF COLUMBIA

EXHIBIT J

CONFIDENTIAL EXHIBIT

FILED UNDER SEAL